March 2013

Intelligent Portable Aerial Surveillance System

Adam L. Blumenau
Worcester Polytechnic Institute

Adrian Charles Sudol
Worcester Polytechnic Institute

Alec Andrew Ishak
Worcester Polytechnic Institute

Brett Louis Limone
Worcester Polytechnic Institute

Corey Wayne Russell
Worcester Polytechnic Institute

See next page for additional authors

Author
Adam L. Blumenau, Adrian Charles Sudol, Alec Andrew Ishak, Brett Louis Limone, Corey Wayne Russell, and Zachary R. Mintz

Project Number: MQP.TP1.WND2

Design and Realization of an Intelligent Portable Aerial Surveillance System (IPASS)

A Major Qualifying Project Report
submitted to the faculty of WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the degree of Bachelor of Science

Submitted by:

Adam Blumenau
Robotics Engineering

Alec Ishak
Robotics Engineering

Brett Limone
Robotics Engineering

Zachary Mintz
Robotics Engineering

Corey Russell
Robotics Engineering

Adrian Sudol
Robotics Engineering

Advisors:

Professor Taskin Padir
Robotics Engineering and Electrical and Computer Engineering

Professor Lifeng Lai
Electrical and Computer Engineering

On
March 12th, 2013

This material is based on research sponsored by Air Force Research Laboratory under agreement FA8650-09-2-7929. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes not withstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsement, either expressed or implied, of Air Force Research Laboratory, the U.S. Government, or OAI.

Abstract

Intelligence, Surveillance, and Reconnaissance (ISR) are critical to military operations. To meet this need, the armed forces have developed a variety of unmanned aerial vehicles (UAV). While UAVs differ wildly in form, cost, and function, they are generally expensive, require several people to setup, operate, and maintain them, and are designed for long range surveillance. This report presents the design and development of a UAV that is inexpensive, one-man operable, and capable of short range surveillance. Based on the requirements provided by the Air Force Research Laboratory, the team established a set of design specifications to guide the design of the UAV. The UAV design is lightweight, durable, has a small form factor when disassembled, displays image data from multiple cameras, and is tele-operated. The system includes an integrated propulsion system, electronics box, and ground station providing a foundation for further advancement in man-portable aerial surveillance systems.

Table of Contents

Table of Figures

Table of Tables

1. Introduction

There is currently the need for an inexpensive and lightweight Unmanned Aerial Vehicle (UAV). This need stems from a high demand for Intelligence, Surveillance, and Reconnaissance (ISR) and the rising cost of UAV systems. The technology is now available to provide ample surveillance in a short time frame; however, this technology is prohibitively expensive for most applications.

An inexpensive solution to this problem would open small scale UAV use and development to a variety of consumers. Inexpensive small scale UAVs could be applied for use in military, law enforcement, rescue operations, and civilian purposes.

The Air Force Research Laboratory (AFRL) is working to address this issue. It is currently funding two project teams to create a solution to this problem. The work of these two teams builds on previous work performed at Wright State University (WSU). The team at Worcester Polytechnic Institute (WPI) has worked towards a solution to this need by utilizing the principles of robotics engineering to develop an inexpensive surveillance UAV with the capability to operate quickly and intelligently.

This report details the work that occurred in order to solve this problem. The report will begin by detailing relevant background information regarding UAVs in military and civilian applications. Problem and design specifications are defined in order to establish the considerations that are addressed in this project. System design and development are then described by detailing the considerations and implementations in development of the systems and components in the IPASS. The results of full system testing are detailed followed by a conclusion of the work done and future work that can be accomplished to further develop the IPASS technology.

1.1 Background

Intelligence gathering is critical to successful military operations. Information about an area as well as ally and enemy locations allows for better and more accurate decisions to be made, increasing survivability of warfighters. One of the primary methods of gathering intelligence is through visual surveillance of an area. The Department of Defense (DoD) defines intelligence as "information and knowledge obtained through observation, investigation, analysis, or understanding." Surveillance; one of the primary methods of obtaining intelligence, is a systematic observation of a subject where as much data as possible is gathered [1].

The DoD is currently working to transform the armed forces into a force suitable to confront 21st century adversaries. The DoD has stated that ISR is a key component to this transformation [1]. One of the newest and most revolutionary ways to conduct surveillance in the 21st century is through the use of UAVs. Implementing a robotic element in surveillance allows personnel to remain in a safer environment while more dangerous scouting operations are conducted by UAVs.

UAVs have revolutionized the field of military aviation in recent years. Many say that unmanned aircraft are the most significant development to happen in the field since the invention of stealth technology. Military UAVs have a vast array of uses and come in all shapes and sizes [2]. The roles that UAVs play are quickly expanding and range from primitive practice targets to

reconnaissance, combat, logistics, research, and even commercial civilian purposes. The use of UAVs in military reconnaissance, however, has perhaps the widest variation.

In addition to military uses, UAVs have many applications in search and rescue. It is crucially important to be prepared for natural disasters when they occur to minimize the number of human casualties. Often after natural disasters the area's infrastructure and roads are so damaged or the weather conditions are so treacherous that they inhibit any search and rescue efforts by traditional means. In a disaster situation, UAVs can be used for information gathering and relaying data to emergency personnel. They can also be used to deliver relief supplies, medical equipment, or communication equipment to people isolated by disasters [3].

UAVs are also useful for law enforcement officials. The UAV provides a much lower cost alternative to manned helicopters. Small UAVs that can fit in a police cruiser can also be deployed faster than a helicopter, allowing for a search effort to be established more rapidly. UAVs can also be used in investigating traffic accidents. The UAVs could map the scene with photos faster than humans on foot without interfering with evidence. In the case of a hazardous materials spill, UAVs can help investigate and respond without putting people in harm's way. A hovering UAV can provide law enforcement with a tactical bird's eye view of any scenario involving a standoff with dangerous suspects. In the ongoing war on drugs UAVs can be used to search for illegal narcotics operations [4].

1.1.1 UAV History

The United States Army, Marine Corps, and Air Force all have their own tiers for classifying their UAVs. These tiers are broken up by size, speed, flight ceiling, and capabilities. The Army's tier 1 is reserved for small, low endurance, low altitude, and possibly hand launched UAVs such as the RQ-11A/B Raven. Tier 2 is for short range, low to medium altitude, long endurance, tactical UAVs such as the RQ-7A/B. The Army's tier 3 encompasses medium to high altitude, long endurance, tactical UAVs like the RQ-5A. The US Marine Corps has a similar set of tiers, but includes an N/A tier comprised of micro UAVs, often called Micro Air Vehicles (MAV).

The classification that the US Air Force uses is slightly different due to the different duties that the Air Force has compared to the Army and Marine Corps. The Air Force tier N/A is for small and micro UAVs. Tier 1 contains low altitude long endurance UAVs. Tier 2 has medium altitude, long endurance UAVs. Tier 2+ is reserved for high altitude, long endurance conventional UAVs. Tier 3 is Air Force exclusive and is reserved for high altitude, long endurance, and low observable UAVs such as the X-47B [5]. Both the United States Navy and Coast Guard currently use UAVs for reconnaissance as well, but have not implemented a classification system. These reconnaissance UAVs vary in their size, weight, communication capabilities, and amount of infrastructure and personnel required to use and maintain.

The higher tiers of reconnaissance UAVs require full runways and can be controlled from across the world using satellite communication infrastructure. The RQ-4 Global Hawk weighs around 15,000 pounds and has a 47 foot wingspan. The UAV requires a crew to maintain and pilot it. It requires a launch recovery element pilot, a mission control element pilot, and a sensor operator [7]. Even mid-size tactical UAVs such as the RQ/MQ-1 Predator and MQ-9 Reaper require a full runway and crew to operate. Currently, the Predator and Reaper drones are launched from forward operating bases, and controlled from the continental United States by a

pilot and a sensor operator [6]. The higher tier UAVs are much more expensive than the lower tier. The RQ-4, as seen in figure 7, costs around 35 million per unit [9]. The MQ-9 Reaper, as seen in figure 6, costs around 37 million dollars per unit [6]. While these UAVs may provide large area ISR, they aren't particularly well suited for transmitting the intelligence directly to the infantry.

Figure 1: MQ-9 Reaper [6]

Figure 2: RQ-4 Global Hawk [7]

The low tier UAVs can be carried and deployed on a squad level, often thrown by hand. These small UAVs are controlled by a personal computer based system. The RQ-11 Raven series, for example, weighs 4.2 pounds, has a 4.5 foot wingspan, can fly for around 80 minutes, and has an effective range of about 6 miles. The UAV is hand launched like a model plane and can either be remote controlled or navigate autonomously by way of set GPS waypoints. The controls are handled by the Ground Control Unit which streams real time video and picture data from the UAV. This class of UAV is useful for ISR in a variety of situations, but can still be relatively expensive. The entire RQ-11 system, as seen in figure 8, costs around $250,000, with each Raven UAV costing around $35,000 [9].

Figure 3: RQ-11 UAV [9]

Currently there are a few MAVs that can be used for squad level ISR that take a different approach than the tier 1 UAVs. The Honeywell T-Hawk MAV is a small squad portable Vertical Take-Off and Landing (VTOL) UAV as seen in figure 9. It has the ability to navigate waypoints and flight plans but also has the unique ability to "hover and stare" [10]. With a range of 6 miles combined with VTOL, the T-Hawk is better suited for urban areas than the small plane like UAVs, but is more expensive.

Figure 4: Honeywell T-Hawk [10]

It is apparent that UAVs of all shapes and sizes are being used by militaries all over the world, especially in the United States. In summary, there are different scales or classes of UAVs for military ISR. There are niches for large world roaming UAVs, mid-size tactical UAVs, and smaller squad deployable UAVs. Currently the smallest VTOL UAV is too large for one person

to carry. That is why there is a need for an even smaller and, more importantly, low cost intelligent portable aerial surveillance system.

1.1.2 Limitations of Current UAVs

There is a variety of different UAVs designed for many different tasks. While most have been designed by the United States military for combat reconnaissance purposes, there are still many factors in which they vary. UAVs can range in size from a few feet to having over ten foot wingspans. They are also designed to run for times ranging from a few hours to two weeks. There are, however, a few commonalities. UAVs are very expensive, with complex drones costing upwards of millions of dollars (not accounting for operating costs or resources). Most established UAVs are also too large to take into the field; they must instead be launched from a runway at a base capable of maintaining them.

Small UAVs are not much more inexpensive than their full sized counterparts and are not designed for squad based operations. For example, the AeroVironment RQ-11 Raven, which is used by not only the United States but more than ten other countries, costs $35,000 for a single unit and $250,000 for a total system. While the UAV can be launched by hand and weighs 1.9 Kg it is not very portable. As can be seen in figure 10, the UAV requires two suitcases to hold its components in addition to another suitcase sized receiver. The RQ-11 Raven is also designed for flights from one to one and a half hours, making it unsuitable for short reconnaissance missions.

Figure 5: The RQ-11 Raven being assembled before flight.
Note the second soldier on top of the jeep setting up the receiver [9].

When looking at the current UAV technology a clear need for an inexpensive man-portable UAV system arises. The large UAV systems, which can cost millions of dollars, can weigh several tons and are only useful for high level reconnaissance missions, not in close quarters. These large systems are not man portable and require a team to operate. Most small UAV systems still take several men to transport and cost thousands of dollars. Making a small UAV system would create many benefits; it would reduce the amount of gear that soldiers would need to carry, be less noticeable to enemy units, and the system could be easily stored for transport. If the system was made small enough for one solider to carry, more than one system

could be transported by a squad to increase the range and decrease the time it takes to obtain surveillance data. The small UAV size would also me more suitable for short range missions; the reason for this is because the smaller size would have a less powerful battery therefore reducing the time the drone could be in flight. Overall, a smaller UAV form factor would be ideal for most urban military missions.

The UAV drones that are used today generally cost thousands to millions of dollars. If that cost could be reduced it would be beneficial to the military. Low cost UAV systems could be more widely deployed for more surveillance coverage on the battle field because the military could afford to deploy more drones. If the cost of a drone could be reduced enough, the drone could be considered disposable. The benefit of a disposable system is that if the UAV were to crash or land in an inhospitable location the UAV could be left behind because it is easy to replace. Making an inexpensive UAV system would also allow for the millions of dollars being spent on UAVs to be allocated to more pressing divisions of the military.

1.1.3 Current Solutions

A number of different small scale aerial imaging devices have emerged with relation to this problem. The Massachusetts Institute of Technology (MIT) has created a device, as seen in figure 11, for “bounce imaging” that captures surveillance data without the aid of flight [29]. While low cost, this device must be physically thrown at the location image data is intended to be captured from, limiting the possibilities for surveillance.

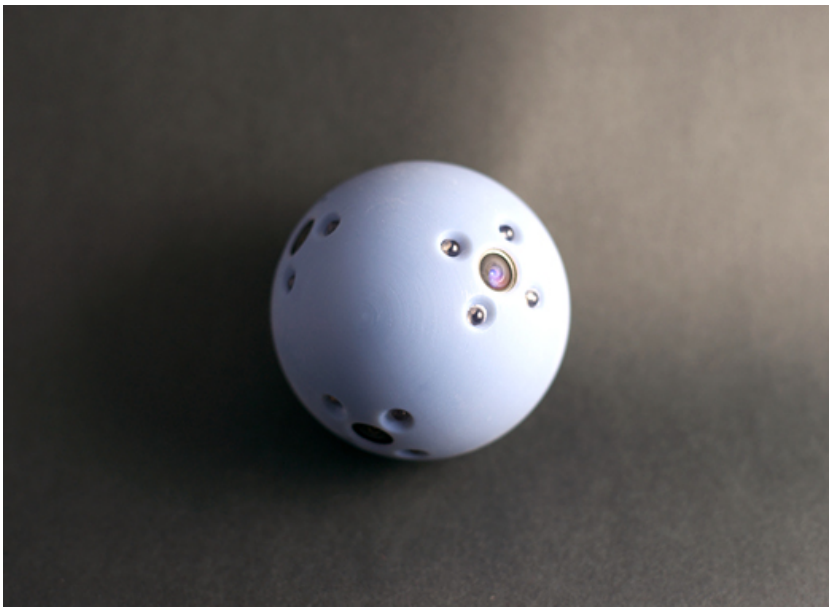


Figure 6: MIT’s bounce imaging device [29]

The senseFly company has developed a small scale UAV designed for quick and easy deployment in order to collect surveillance data of the surrounding area as seen in figure 12. The eBee UAV can intelligently plan flight paths before flight and return to the user at the end of its flight, but is prohibitively expensive [30].

Figure 7: The eBee 3D mapping UAV [30]

Prox Dynamics has developed a small scale surveillance UAV for military applications. The PD 100 PRS UAV is single man portable and man operable as seen in figure 13. While this UAV's small form factor makes it easily portable, a set of 160 units costs £20 million (about $30 million) at a minimum [33].

Figure 8: The PD 100 PRS in the field [33]

Fumiyuki Sato has invented a flying sphere, as seen in figure 14, for 110,000 yen ($1390) that is capable of performing complex aerial maneuvers. The sphere has an approximate eight minute flight time and in the event of a loss of power or unexpected collision it is protected by its spherical outer frame [29].

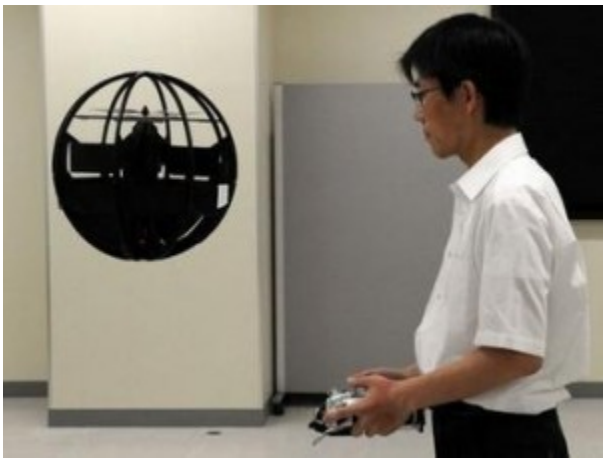


Figure 9: Fumiyuki Sato's Flying Ball **[**29]

Lite Machines has developed an unconventional UAV design that utilizes coaxial rotors to reduce complexity. The UAV, as seen in figure 15, is designed to be portable and easily launchable [28].

Figure 10: Lite Machines Voyeur UAV [28]

1.2 Societal Impact

Access to lightweight and inexpensive aerial surveillance systems has the potential to change the way mapping and navigation is accomplished. An inexpensive series of UAVs would give civilians access to technology that, as of now, is primarily used in military applications. Scientists could use small, short-range UAVs to capture visual data in a variety of different environments which could provide useful information in the fields of geology, biology, meteorology, and environmental sciences. Civil engineers and architects could use small scale UAVs to survey an area before construction and check the safety of the site while the building is undergoing construction.

Inexpensive UAVs have significant potential in rescue operations. In areas where the environment may be unsafe or unstable, such as a forest fire or hurricane, having a low-cost UAV that is optionally-recoverable would be highly beneficial. The UAV would be able to

achieve its goal of launching into the air and providing visual data and if it was lost during operation there would be no significant strain on resources to purchase another.

In the event of extensive proliferation of inexpensive and disposable UAVs, the environmental and societal impact must be considered. The components used in the IPASS are potentially hazardous and could cause negative environmental changes should a number of systems remain unrecovered. To combat this potential danger, a GPS receiver has been implemented, but not integrated, in order to provide location data of the IPASS so that it can be more easily recovered. The privacy issues inherent in wide scale availability of inexpensive surveillance technology are also of concern both in the professional and amateur field. However, this is not a novel concern with preexisting technologies such as Google Earth, cell phone GPS tracking, and personal information security with regards to social networks. The IPASS is designed for the protection of war fighters and to provide useful surveillance data for beneficial applications. While effort has been made to minimize the negative impact of this technology, it is the users who can determine if its application is safe and beneficial.

1.3 Project Description

The AFRL is working on developing a surveillance ordinance to improve upon current un-manned surveillance methods. The project team believes that a robotic solution is capable of fulfilling various surveillance requirements while remaining inexpensive and compact. The AFRL Sensors Directorate has previously conducted this project with a student team at WSU as part of its AFRL Student Challenge. This challenge was developed to address issues in the disciplines of aerospace, mechanical, electrical, computer, and software engineering and allow students the opportunity to work with the AFRL. This student challenge serves to tackle "specific technical challenges to AFRL" while meeting college senior capstone requirements [31].

This project was developed simultaneously by two teams; one team represented WPI while the second represented WSU. The two teams communicated and collaborated but developed two designs independently. The main goal of the WPI team is to design, develop, and test an unmanned aerial vehicle to serve as a surveillance unit for a user on the ground. The design focuses on integrating features that are believed to be important for vertical flight and aerial surveillance. Based on AFRL's goals for the surveillance ordinance, it was possible to separate the goals into three main categories.

1. Takeoff and landing
2. Gathering and sending image data
3. Gathering and sending location data

To assist in the research and development associated with implementing this solution, the AFRL funded the IPASS project through the Ohio Aerospace Institute (OAI). The team drew up a preliminary outline of expected expenses prior to doing any spending on the project. Table 1 shows the estimated costs associated with developing a solution to meet these requirements.

Table 1: Project Budget

Item	Cost
Ground Station Laptop	$800
Mechanical Systems including materials and manufacturing.	$2000
Cameras and Lenses	$600
Computing System	$500
Sensors	$500
Batteries and Charger	$300
Communication System	$400
Travel cost for demonstration	$2000
Miscellaneous	$900
Total	**$8000**

1.4 Design Specifications

The AFRL was broad in its design requirements and was willing to allow the team to make adjustments based on feasibility and relation to robotic applications. Because of this flexibility, the team added several additional design requirements to better address the team's goals for the project. The first five requirements in the following list were specified by the AFRL while the rest were the team's initial design goals.

1. The system must reach heights of at least 100 ft.
2. The system must survive a fall of 30 ft. unassisted
3. The system must be lighter than 20 lbs.
4. The system must have a device to retard its fall.
5. The system must have a vision system to provide a wide field of view of the surrounding area while airborne.
6. The system must be able to sense its location.
7. The system must have an embedded computing system for image processing, sensor fusion, actuation, communications and control.
8. The system must be able to transmit visual and location data while airborne to a user on the ground.
9. The system must be able to transmit data wirelessly to a ground station with a downlink rage of up to 200 ft.
10. The system must be user friendly and easy to operate.

The 100 ft. minimum flight height is a design requirement for two main reasons. The AFRL felt that the system should be able to take useful pictures from 100 ft. in the air. Secondly, 100 ft. is a sufficient height to clear most obstacles that would normally block vision.

The system should be able to survive a 30 ft. drop as a fail-safe. In the event of a system failure the user should still be able to retrieve the system for reuse. In order to ensure consistent survivability, the system should be able to slow its descent.

In order for the system to be transported and launched easily, the system should be lightweight. If the user of the system is already carrying another load, carrying this system should not impact the user significantly.

The on-board vision system should be able to take several series of pictures and transmit them to the user during the system's operation. Once the images are sent to the user the ground station will stitch the pictures together into a single image. These images ware saved to the ground station and displayed. In additional to visual data, the team decided that location data is also useful to the user. On-board sensors will transmit location data to the user.

The range of the wireless communications should be larger than the minimum flight height of the system. This greater range will ensure constant connectivity with the system and allow for lateral movement of the IPASS and user during operation.

While there was no specified cost requirement, the goal of the project is to create an inexpensive system. A low production cost will allow for each individual system to be easily replaced as necessary.

1.5 Team Organization

The six member project team was subdivided to better accomplish the IPASS design requirements. On the highest level, team members were divided to work on the body design, launch and descent or embedded computing, vision system, sensing, and software. As the project progressed however, there was a much greater overlap in member responsibilities. Table 3 shows what aspects of the project each team member was accountable for. All team members were expected to have at least a basic a running knowledge about every aspect of the IPASS while only a few needed an in depth knowledge.

Table 2: IPASS member responsibility

Responsibility	Primarily Responsible	Secondarily Responsible	Knowledgeable
Chassis Design	Limone	Blumenau	Ishak, Mintz, Russell, Sudol
Propulsion System Design	Blumenau	Mintz	Ishak, Limone, Russell, Sudol
Launch Testing	Mintz	Blumenau	Ishak, Limone, Russell, Sudol
Vision System Design	Ishak	Sudol	Blumenau, Limone, Mintz, Russell
Image Processing	Limone	Russell	Blumenau, Ishak, Mintz, Sudol
Embedded Computing	Sudol	Ishak	Blumenau, Limone, Mintz, Russell
Location Sensing	Blumenau	Russell	Ishak, Limone, Mintz, Sudol
Communication Implementation	Sudol	Russell	Blumenau, Ishak, Limone, Mintz
Ground Station GUI	Russell	Sudol	Blumenau, Ishak, Limone, Mintz

1.6 Project Timeline

The following weeks detail project milestones that were accomplished over the course of twenty one weeks. Each date coincides with a weekly meeting that occurred with the project sponsor, Richard Van Hook. The project spanned from to August 23, 2012 to March 12, 2013. Key milestones are indicated in bold.

Weeks 1-5

1. Operational SCP server
2. Document templates designed
3. Primary camera choice : TTL serial JPEG with NTSC video
4. Raspberry Pi and Arduino selected as controllers
5. **Primary mechanical design made**

Week 6

1. Began IMU research
2. Began GPS research
3. Determined bandwidth properties
4. Found motors and propellers to test
5. Materials applied to mechanical design

Week 7

1. Prototype chassis cut from laser cutter
2. Xbee speed tests complete
3. Rotors, motors, IMUs, and ESCs received in mail, testing to begin
4. Begin writing formal project paper

Week 8

1. Retrieving IMU data
2. GPS received in mail
3. Progress on paper
4. Testing with parts received last week

Week 9

1. Began work on final paper: introduction, background, design requirements, and specifications
2. Interfacing cameras
3. Retrieving data from IMU
4. Retrieving data from GPS
5. Controller PPM signal operational
6. **Second prototype designed**

Week 10

1. IMU data filtered
2. second prototype design fabricated

3. GPS tested

Week 11

1. **First launch test**
2. Electronics box manufactured
3. GPS data parsed

Week 12

1. **Third iteration designed**
2. **USB camera and SPI camera images received**
3. **Image stitched with JavaCV**

Weeks 13 & 14

1. Multiple thrust tests completed
2. New propellers and motors acquired and installed
3. IPASS design updated
4. Communication with cameras
5. **Mid project evaluation presentation**
6. WSU team + sponsor visit
7. WSU team collaboration

Week 15

1. **Drop tests to evaluate design**
2. **Thrust tests**

Week 16

1. Subsequent launch tests
2. **Pictures from Caspa cameras**

Week 17

1. Camera change to USB cameras, tested and functional
2. Data transfer to and from Gumstix
3. **Wireless communications established**
4. GUI front end complete
5. Launch tests

Week 18

1. **Image data wirelessly transmitted**
2. Message protocol established
3. **Fourth prototype designed**
4. Begin system integration

Week 19

1. Launch tests
2. System integration progress

Week 20

1. Electronics box assembled, debugging to be done
2. **Full wireless communications established between system and ground station**
3. Image stitching approved
4. Pelican case ordered to store IPASS

Week 21

1. **Full system tests: image capture, launch, and drop tests**

2. System Design and Development

This chapter details the design considerations and implementations made in the development of the IPASS. The proposed solution is described in order to frame the design choices made with regards to the propulsion, chassis, power system, embedded computing, vision system, onboard sensing, data transfer, software, and ground station of the IPASS. Each subsection details the design considerations and implementation of the relevant subsystem.

2.1 Proposed Solution

In order to meet the project goals the team developed an Intelligent Portable Aerial Surveillance System (IPASS). To satisfy this acronym, the IPASS is designed to intelligently travel to a 100 foot elevation, to be single man portable, to send aerial image data to the user for surveillance purposes. This system is broken into three major components: chassis, electronics box, and ground station. The system is designed to be transported into a dangerous environment, assembled quickly, and launched vertically. While airborne, images are captured and sent to a ground station. This ground station presents all relevant data to the user. Figure 16 presents the final design of the IPASS.

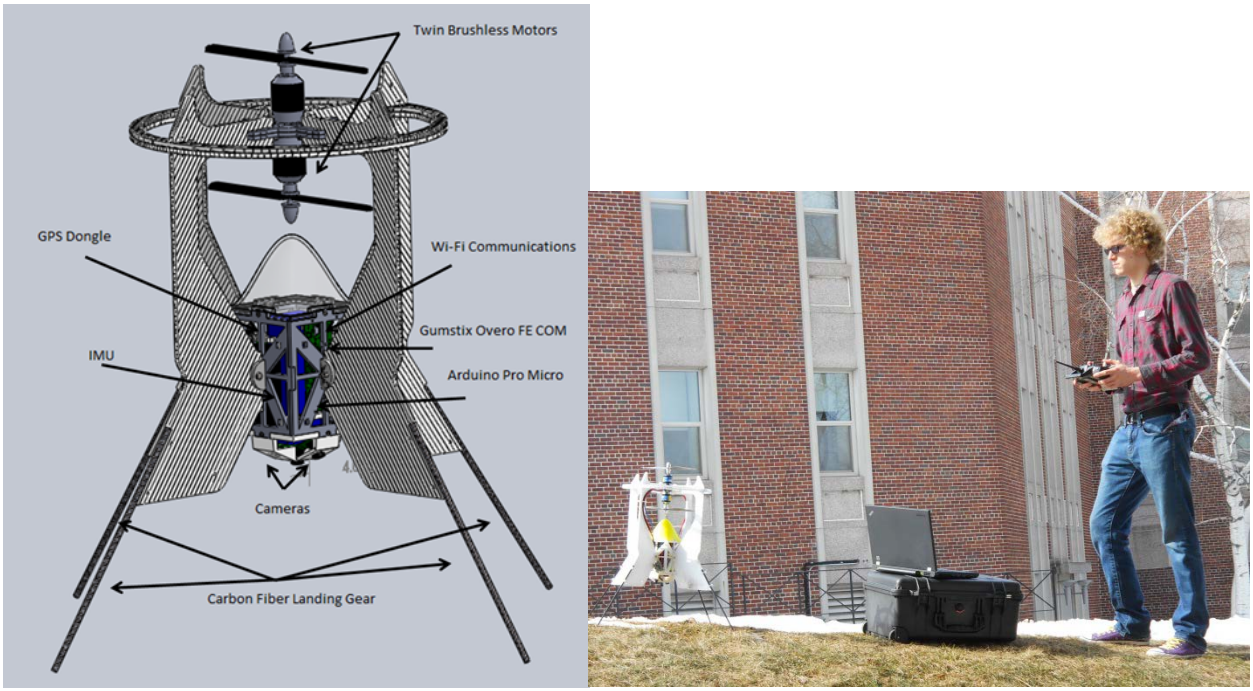


Figure 11: Final design of the IPASS and realized UAV

The final IPASS design contains two brushless DC motor driven propellers mounted coaxially, a propeller protection ring, four quarter frames to mount the carbon fiber landing gear, and an electronics box. The electronics box contains a LiFePo battery, two electronic speed controllers, an Arduino Pro Micro, a Gumstix Overo, a radio receiver, and three USB cameras connected to a USB hub.

2.2 Propulsion Considerations

There are many different options for propelling the IPASS into the air, including cannons, catapults, balloons, rockets, and propellers. Each option can be evaluated based on various specifications including cost, mobility, safety, complexity, and overall feasibility as part of meeting the system requirements. Previous work conducted by the WSU team was considered in the decision making process regarding the propulsion system.

2.2.1 WSU Considerations

The previous WSU team that attempted this project researched pneumatic launcher cannons, as seen in figure 17, to launch the device into the air. This required a large launcher made of PVC pipe fashioned with a pressure tank and a barrel. The pressure tank would need to be filled with compressed air from another source such as a large air compressor. These are commonly used for supplying air to pneumatic tools. With this system, the WSU team determined through testing that it would require too much pressure to be feasible to launch the device. This system would also be too large and heavy to be man portable and would require a power source to prime the pressure chamber with air in the field [11].

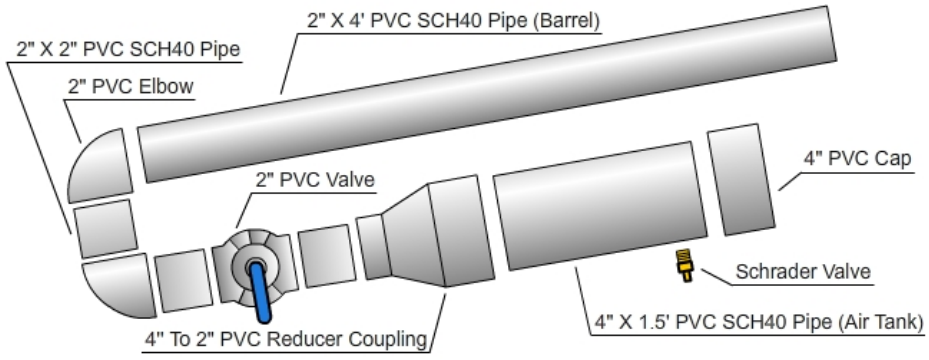


Figure 12: WSU pneumatic launcher design **[11]**

The WSU team researched the use of an elastic catapult. This was fashioned as a large upward facing slingshot that used bungee cords to provide a spring force intended to launch the device vertically. The WSU team determined that this method would be feasible to launch the device vertically to the desired elevation, but the amount of force required to pull back the slingshot was too great for humans to apply quickly in the field. During testing, a tractor was needed to pull the catapult to full potential energy. This method also would require the use of a large catapult which means that the entire system is no longer single man portable [11].

The third option that the previous team had considered was using a large weather balloon to lift the device into the air. The team considered the weight of the device and the required tethers to determine how large a balloon would be needed. It was determined that a 10 foot diameter weather balloon would be needed to attain the desired elevation. This required a large can of helium to fill the balloon. The materials and effort needed to launch this balloon make this unsuitable for a man portable system [11].

2.2.2 Propulsion Design Considerations

After taking into account the efforts of the previous team, it was apparent that none of these aforementioned methods would be feasible for the system. As seen in Table 4, the ease of

use, set up time, complexity, cost, and portability of the previous team's design solutions were evaluated for their feasibility.

Table 3: Previous team's design solutions analysis

	Pneumatic Cannon	Elastic Catapult	Weather Balloon
Can this method achieve the desired elevation?	Yes	Yes	Yes
Is this method single man Portable?	No (with compressed air source)	No	No (when inflated)
Does this method have a low cost per use?	Yes	Yes	No
Can this method be deployed quickly?	Yes	No	No

Consideration: Rockets

A common method of small scale propulsion is model rocketry. The engines, as seen in figure 19, for model rockets come in a variety of sizes, thrust levels, and are relatively inexpensive. Research was done into the kinds of rockets available based on their specifications. Newtonian physics allows approximations on how high the rocket could launch a given object. The height was approximated by taking the average thrust in Newtons, the mass of the object in Kg, and the burn time of the motor. The basic Matlab feasibility calculation can be viewed in figure 18. Using this simulation it was determined that rocket engines would be a feasible method for providing the force required to launch the IPASS.

```
%%inputs
mass=3;%in kg
avgthrust=34;%% avg thrust of motor in N, form datasheet
burntime=1.7;%% in s

yburn=.5*(avgthrust/mass)*(burntime)^2;
vburn=(avgthrust/mass)*burntime;

%output in meters
maxheight=-.5*9.8*(vburn/9.8)^2+vburn*(vburn/9.8)+yburn;
```

Figure 13: MATLAB simulation of thrust

Rocket motors typically use some kind of electronic ignition and require a launch pad with a vertical guide rod. This setup may make this no longer a man portable system. For a rocket to work, it needs to be aerodynamic, and have a small form factor which limits component choice.

Another downside to this technology is that the large rocket engines needed to lift the system are regulated at a federal level. Hobby rockets expel hot gasses from burning solid fuel making them dangerous to use in urban environments. The high heat output and flammable nature of the rocket motors pose a danger to the internal components.

Figure 14: Model Rocket Engine [12]

Consideration: Electric Motors

The feasibility of using a brushless dc motor can be evidenced by approximating the thrust generated when attached to a propeller, and comparing it to what the system might weigh based on the required components. The static thrust of a brushless dc motor and propeller can be approximated the using Equation 1: Brushless dc motor thrust approximation [13]:

$$Thrust = [(\eta * P)^2 * 2\pi R^2 * \rho]^{1/3}$$

Equation 1: Brushless dc motor thrust approximation

Where η is the propeller hover efficiency which depends on the pitch and width of the propeller, P is the shaft power, R is the radius of the propeller in meters, and ρ is the density of the air.

Thrust is approximated for an 800 watt brushless dc motor below. Standard propeller hover efficiency is 0.8 for hobby propellers and a 10 inch diameter (0.127m radius) is a standard propeller size. The density of air is approximately 1.22 kg/m^3. If these values are input into Equation 1: Brushless dc motor thrust approximation the output thrust generated in Newtons or Kilogram force can be useful in approximating the acceleration of the IPASS.

$$Thrust = \left[(.8 * 800W)^2 * 2\pi(.127)^2 * 1.22\left(\frac{kg}{m^3}\right)\right]^{1/3} \cong 37N \cong 3.8\,KgF$$

If two motors are used, then this thrust should double and the system will output approximately 7.6 KgF or 74.5N of downward thrust. If the IPASS weighs 2Kg, or 19.6 N, then:

$$\sum F = mA = 74.5 - 19.6 = 55N$$

Thus the IPASS would accelerate upward at 27m/s^2. This approximation proved that using brushless dc motors and propellers was a feasible option.

2.2.3 Descent Considerations

Using many of the above methods the system can achieve the upward force required to launch the IPASS; unfortunately most of these methods would require a separate means for retarding the descent of the system. Ideally, this landing will not end in a crash where the core

components of the system are damaged and non-recoverable. A robust descent mechanism is critical to preventing a damaging crash. Multiple descent methods were researched including parachutes, a rotary wing, and propellers.

Consideration: Parachute

A parachute allows the system to slow its descent by using a large, usually round, canopy attached to the chassis using string or rope. The parachute could remain folded inside or on the chassis until it is deployed either using a secondary charge of a rocket or some kind of electromechanical actuation. A problem with using a parachute is that it takes up a significant amount of space and weight, even when folded up. Another limitation is that when the system is descending, there will be sway due to the wind which may affect the quality of the images collected [18].

Consideration: Rotary Wing

The possibility of using a rotary wing design, much like a maple seed, as seen in figure 21, was researched. In this design the rotation of a single wing craft could propel the system upward and control its descent. This would be similar to the Lockheed Martin Samurai flyer, which can be seen in figure 20 [34].

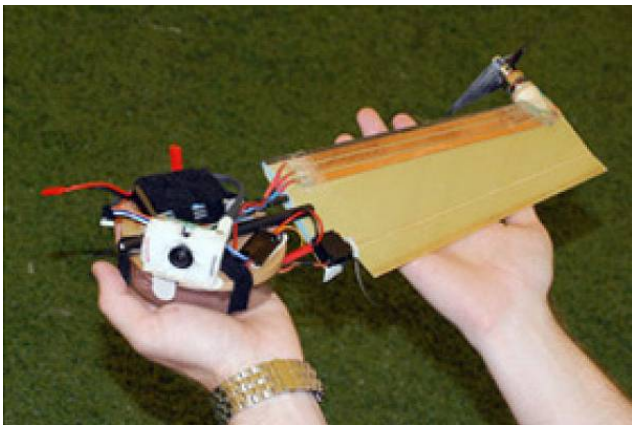


Figure 15: Lockheed Martin Samurai [34]

The team decided against this idea due to its complexity. This would require extensive vision processing to stabilize the resulting images as well as a more complex overall design. With regards to descent, NASA's Jet Propulsion Laboratory conducted research on using droppable maple seed sensors that would act as miniature autogyros which use autorotation to slow descent [15].

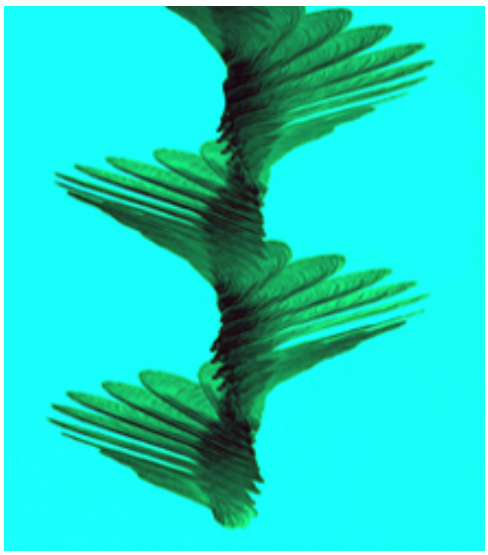


Figure 16: Images of a maple seed falling
[15]

Consideration: Propellers

Another option was to design the IPASS such that it uses passive rotation of propellers to slow its descent. This could be done using a fan like rotor with enough surface area and a shallow pitch to auto-rotate. Autorotation is when the rotation of the rotor caused by air passing through it causes lift [29]. It is commonly used with helicopters in emergency situations as well as in auto-gyros. This method could possibly slow down the descent of the IPASS, but inducing a rotation to the system could affect image quality due to the amount of time it takes for the sensor to capture an image.

The final option considered was to use powered propellers to slow the descent of the IPASS. If the system was launched using the propellers to provide thrust, the thrust could be reduced at the apex of its trajectory to enable the system to descend. By using brushless dc motors to both launch and control the fall of the chassis, required space, and complexity can be reduced. The team decided to use powered propellers for both launch and descent because of the safety, simplicity, and ease of implementation. Later during the design process, the team used a Matlab simulation to aid in diagnosing launch issues. In this simulation the IPASS flight path was simulated in one, two, and three dimensions. Simulating the flight path in one dimension (up and down) showed that based on the thrust calculations and weight of the system the IPASS would launch normally as can be seen in figure 22. In simulating two dimensions (up and down, and pitch) it was shown that based on the weight distribution and resultant center of mass of the system would not ascend directly upward as can be seen in figure 23. When simulating the IPASS in three dimensions (up and down, pitch, and rotation) the flight of the system could be stabilized by applying a rotation about the axis of the propellers as can be seen in figure 24. This rotation would later be applied in the system by running the motors in a co-rotating fashion. Full details of this simulation can be seen in Appendix G.

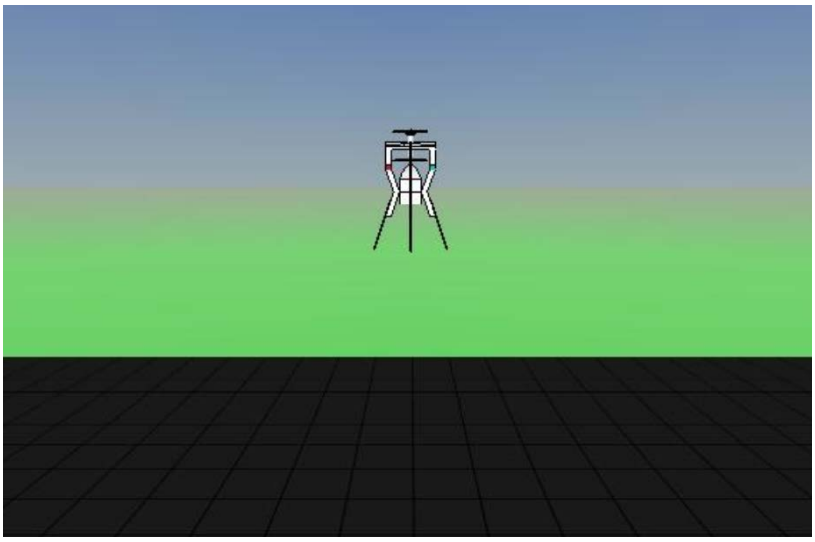


Figure 17: Simulation of IPASS flight in one dimension

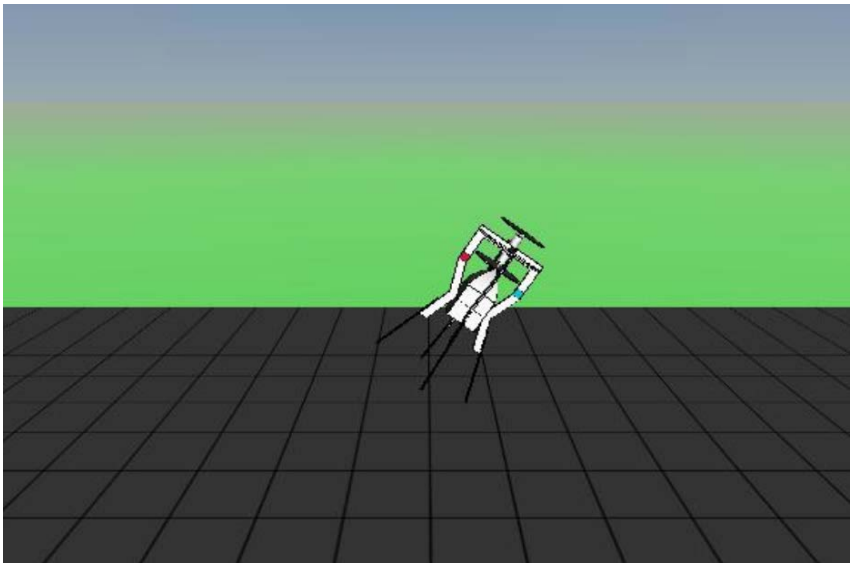


Figure 18: Simulation of IPASS in two dimensions where imbalance is discovered

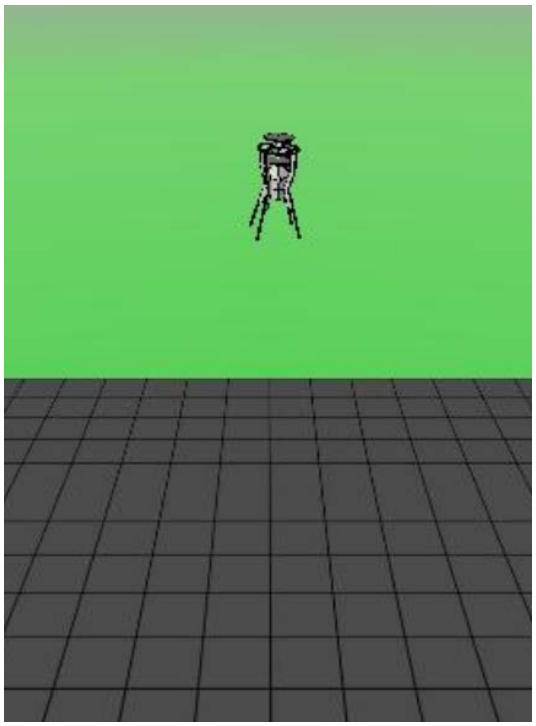


Figure 19: IPASS 3D simulation in which rotation is applied to stabilize flight

2.2.4 Implementation: Propulsion

The following design decisions are the final implementations in the IPASS. These implementations were made based on research and experimentation performed with the previously described propulsion systems.

Motors

The selection of motors was one of the most critical decisions for the propulsion system. The first motors that were selected for use in powering the IPASS's propulsion were the NTM Prop Drive Series 35-30A 1400kV. These motors draw up to 35A at full power and run at a voltage of 13.2 V. It was determined that for the propellers that were selected the static thrust should be 1.5kg for a motor-propeller pair. During thrust tests of the IPASS it was determined that the motors did not generate enough lift to get the system airborne. After more research was done, the NTM motors were replaced with the E-Flite Power 25 Brushless Outrunner Motors. These motors are capable of drawing 58A burst current and run at 1250kV. The E-Flite motors generate sufficient lift for the IPASS to launch itself off the ground.

Electronic Speed Controllers

The ESCs were a critical component for the IPASS as they are responsible driving the motors. The ESCs that are used on the IPASS are the Turnigy Thrust 55A SBEC Brushless Speed Controllers. These ESCs were chosen in order to meet the requirements of the motors. These ESCs can handle up to 55A of sustained current and also provided a battery elimination circuit (BEC) which provides 5V power to other components on the IPASS. These specific ESCs were chosen because they have the appropriate capability for driving the selected motors.

Propellers

The propellers for the IPASS are important in that they provide the means to convert the motors' rotations into thrust. To meet IPASS design requirements, the propellers needed to be lightweight and disposable. The propellers that were chosen for the IPASS are the APC 1047 Slow Flyer Props. These propellers come as a counter rotating pair. The APC propellers seen in figure 25 are ten inches in diameter with a 4.7 inch pitch.

Figure 20: Propeller pair used in the IPASS

Radio Control

The IPASS's motors are radio controlled (RC). The RC control is used because the Federal Aviation Administration (FAA) guidelines state that there must always be a human in the loop when operating a UAV [32]. The motor speed is controlled by a human operator to maintain safety during operation. The human operator controls the system by adjusting motor speed using a joystick.

2.3 Chassis

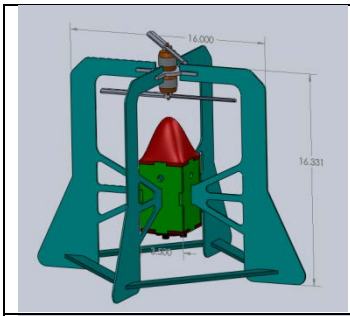
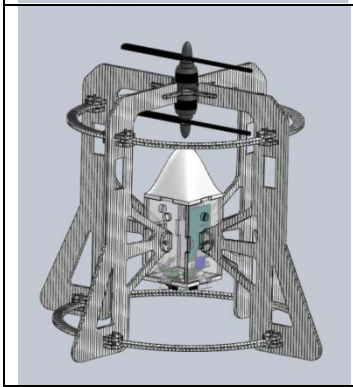
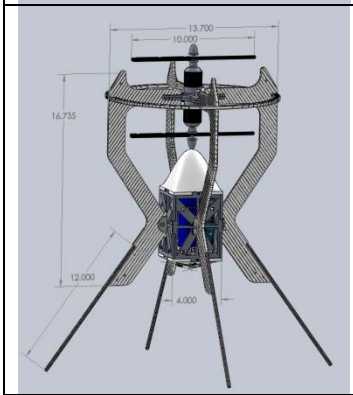
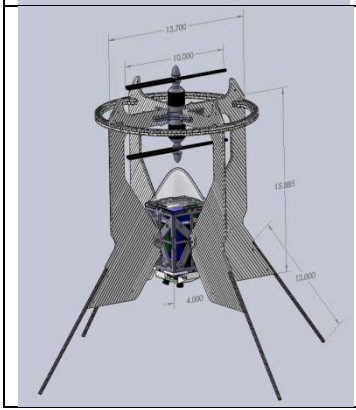
The mechanical design of the IPASS is developed to meet the functional requirements of the project. Some aspects of the design were influenced by existing UAVs and vertical takeoff and landing aircraft such as a disposable outer chassis for shock absorption, a low center of mass for stability, and coaxial propellers. These existing craft included model rockets, military vehicles, existing military UAVs, civilian UAVs and hobby craft. Effort iss made to use commercial off-the-shelf parts and materials where possible to keep the cost low.

2.3.1 Chassis Design

The IPASS is designed to be stored inside a man portable case until needed. The chassis is designed to be assembled and disassembled in the field without the need for tools. This means using different fastening methods such as pins and tie wraps as opposed to nuts, bolts, and screws. The chassis is required to survive a 30 foot drop. It was decided that the IPASS would be designed such that the outer chassis would be made of a low cost material. A low cost outer shell allows for an optionally recoverable chassis. The box that holds and protects the electronics is designed to be recoverable to allow the system to remain cost effective.

There were four major revisions of the chassis design throughout the project.

Table 4: Design iterations

	The first design was developed as a proof of concept, and a drop test platform.
	The second design was developed mostly for testing the chassis material choices, and overall feasibility of the motor mount, electronic box, and outer frame design.
	The third design was intended to be lighter, just as resistant to drops, and easier to rotate in the air.
	The fourth design was intended to fly vertically in a more stable manner, and to perform better at protecting the propellers.

Motor Mounting

After deciding to use brushless dc motors and propellers for propulsion a chassis was designed to properly mount the motors and propellers. The chassis is designed to protect the core electronics, cameras, sensors, and to keep the IPASS traveling in a straight trajectory. The two rotors are contra-rotating such that the torques induced from kinetic energy will cancel each other out, arresting rotation as can be seen in figure 27. If an outside disturbance causes a rotation it can be controlled while in flight by differentially adjusting the speeds of each rotor. The motors will be mounted on the same plate so that the axles are collinear. This phenomenon can be proved using the formula for rotational kinetic energy seen in Equation 2: Rotational kinetic energy.

$$E_{rotational} = \frac{1}{2} I \omega^2$$

Equation 2: Rotational kinetic energy

Where the value I is the moment of inertia of the propeller plus the moment of inertia of the brushless dc motor housing, and ω is the angular velocity of the motor. If I and ω are equal then the kinetic energy about the common axis of rotation would be zero, assuming that the moments of inertia of the propellers and motors are identical with respect to their mutual axis.

$$\sum E_{rotational} = \frac{1}{2} I {\omega_1}^2 - \frac{1}{2} I {\omega_2}^2 = 0$$

Theoretically, the rotation of the IPASS in the air about its axis of rotation can be controlled by varying the ω of one or both of the motors. The yaw velocity control system for this method is modeled in figure 26.

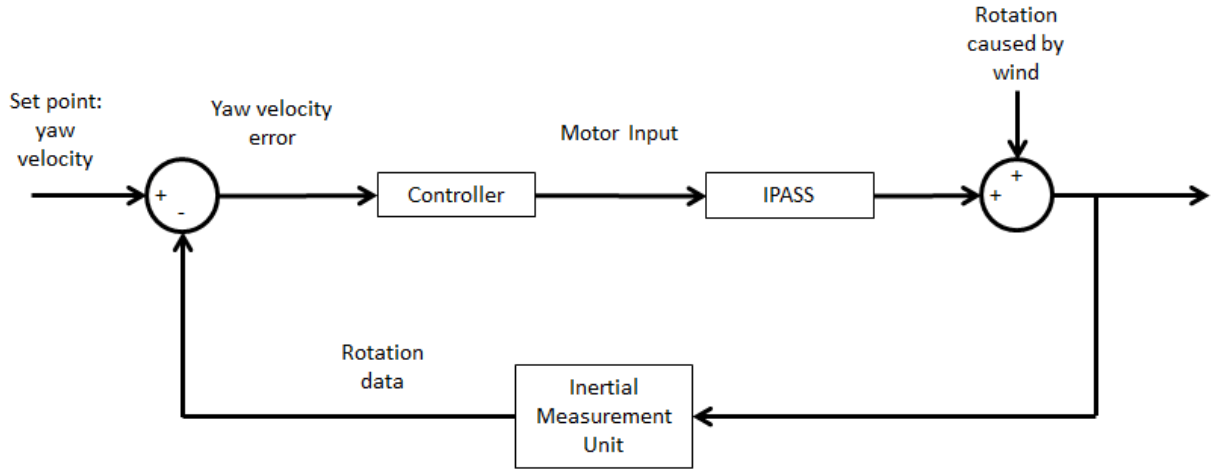


Figure 21: IPASS yaw velocity control system

The motors are mounted on a laser cut piece of Delrin plastic, chosen for its durability. After attempting to use acrylic, it was discovered that the torque produced when one of the propellers collided with an object caused the acrylic to shatter.

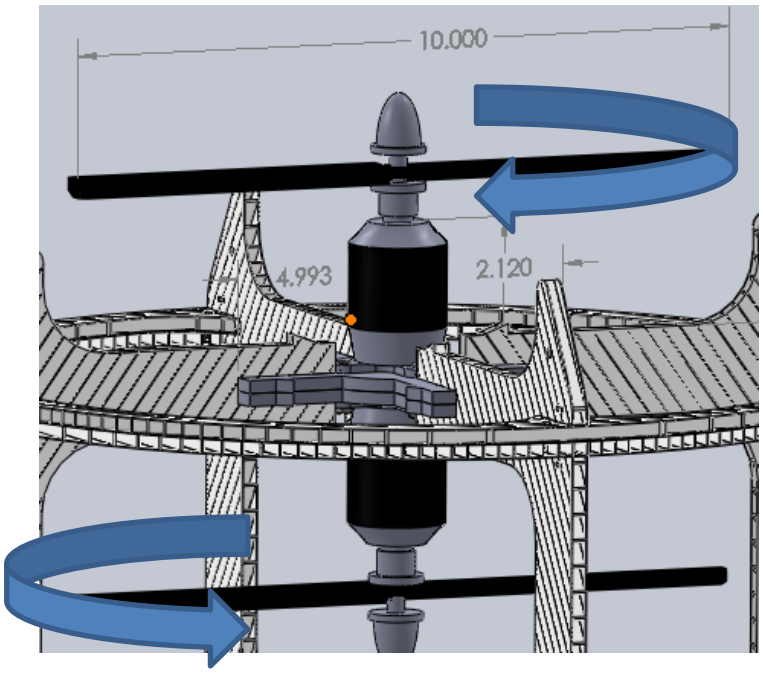


Figure 22: IPASS motor mounting scheme

Frame Material Choice

The material for the IPASS frame had to be light and durable so as to not inhibit takeoff or landing. The choice of material largely depends on the propulsion design choice and available manufacturing methods. When considering a propeller driven option, tough plastic materials were researched. The plastic needs to be durable enough so that the chassis can withstand a 30 ft. drop without damaging the internal electronic components.

The team's primary means of manufacturing plastic components is a laser cutter. The laser cutter is capable of accurately cutting many plastic components. Acrylic was first considered because of the team's familiarity with its material properties. Acrylic, however, is too brittle and not suitable for withstanding impact. Polycarbonates, which are less brittle than acrylic, cannot be cut with the laser cutter. The team came to the conclusion that the laser cut solid sheets of plastic might be too heavy to be lifted by means of electric motors.

After researching solid plastics, a low cost corrugated plastic called Coroplast, as seen in figure 29, made of a polypropylene copolymer was discovered. This material can be laser cut and is lightweight due to corrugation. Testing, detailed in Appendix D, shows that the material is durable, but when too much force is applied it tends to crumple and compress. Results of this these tests can be seen in figure 28 in which the Coroplast material sustained only minor abrasions from a 30 ft. drop. This is ideal for the chassis because the crumpling reduces the impulse from impact to protect the core electronics components. This is similar to how bumpers of modern automobiles are designed to prevent the passengers from absorbing the impact of a collision.

Figure 23: Details of chassis damage after drop test

Figure 24: Coroplast [17]

The material was low cost enough that having a disposable chassis would be cost effective. An interesting phenomenon caused by the corrugation allows the design to take advantage of the way the material resists bending in certain directions. By choosing the orientation in which the parts are cut, the ability of the chassis to resist impact can be optimized.

For intricate parts such as the mounting components and aerodynamic elements of the electronics box, another manufacturing method was required. For complex three dimensional parts a rapid prototyping machine was to be used. The ABS plastic material which comes out of the printer is durable and lightweight because of its porosity. This material is suitable for aerodynamic components and camera mounts. A more durable plastic is used to protect the parts of the system that are intended for reuse.

Outer Frame Design

The outer frame protects an electronics box which contains the embedded computing system, cameras, sensors, and the power system. The design iss inspired by Fumiyuki Sato's flying ball UAV shown in Figure 14. While the IPASS does not perform like the flying ball, it has an outer chassis to protect the rotors and electronics on impact which influenced the design of the IPASS. The Light Machines Voyeur UAV, shown in Figure 15, also influenced the design of the IPASS. It uses contra rotating rotors to control the elevation and rotation of the IPASS, and features a center of mass below the mounted propellers.

Iteration One

The original chassis layout, as seen in figure 30, was designed as a motor mounting plate and an electronics box that would be joined using a protective outer frame. The electronics box must be durable to secure and protect the embedded computing, sensors, cameras, and power system. The size of the original electronics box was chosen to fit a standard Lithium Polymer (Li-Po) battery and a Raspberry Pi, which were the original components. Additional outer frame material that extends below the electronics box protects the cameras from direct impact. The outer frame members which attach to the electronics box are designed to act as shock absorbers when the chassis impacts the ground, suspending the electronics box and absorbing shock and vibration.

In this design the point at which the motors and propellers apply the force to the IPASS was at the motor mounting plate directly above the electronics box. This was chosen so that the IPASS remains upright as it launches. There is an aerodynamic cone on the top of the electronics box to deflect the air.

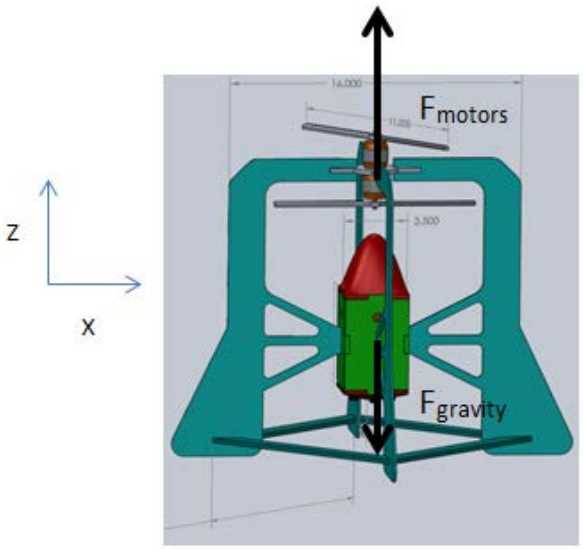


Figure 25: Original IPASS prototype design

The center of mass of the IPASS is designed to be inside the electronics box below the point where the force is applied. This design was developed and verified using SolidWorks. This allows the system to stabilize passively using gravity. If the chassis tips then a component of the force due to gravity will cause rotation so that the thrust applied by the motors is in the vertical

direction. The angle of attack determines how much force is applied perpendicular to the force applied by the motors to rotate the system. The angle of attack increases proportionally with the component of gravitational force perpendicular to the motor force as shown in figure 31.

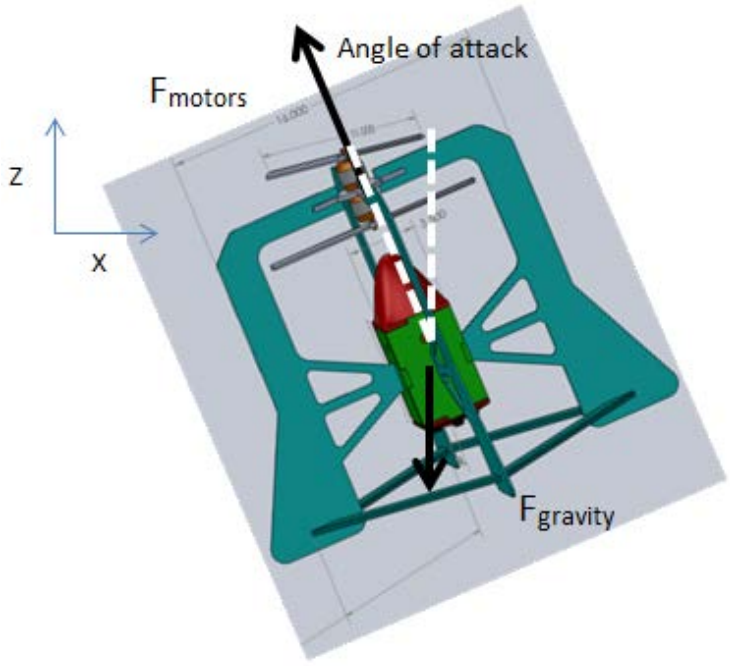


Figure 26: Passive stabilization if the IPASS

The choice of material and basic design were tested by constructing an early prototype chassis, as seen in figure 32, and dropping it from approximately 30 feet. The results showed that the outer chassis sustained some damage including cracks and deformities in the plastic, crumpling on impact absorbing the impulse. This damage can be seen in figure 33. This outer chassis protected the electronics box which sustained minor abrasions. The decision to use the Coroplast for the outer frame of the IPASS was validated by this test. Full results of this test can be found in Appendix D: IPASS Drop Tests.

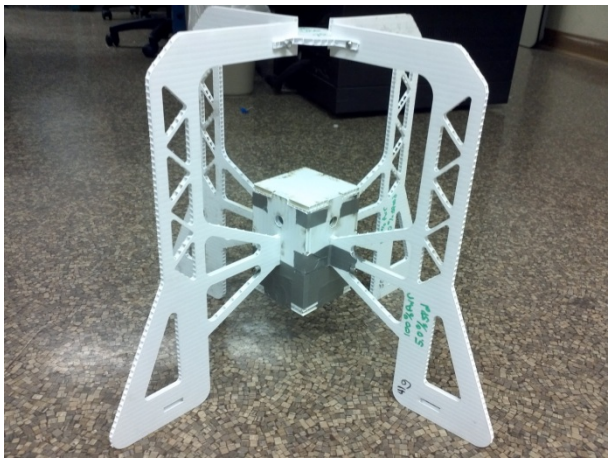


Figure 27: IPASS prototype before drop test

Figure 28: Close up of damage after impact, notice the cracked plastic in the circle

Iteration Two

The next iteration of the design was updated with additional propeller protection. The chassis was updated and cut in a different orientation because it was determined that the Coroplast was better at absorbing impact when the force is directed along the corrugation. It was also determined that Coroplast is too flexible when cut thin and that the acrylic support mounts add too much weight and complexity to the system. The updated design is shown in figure 34 and its mass properties in figure 35.

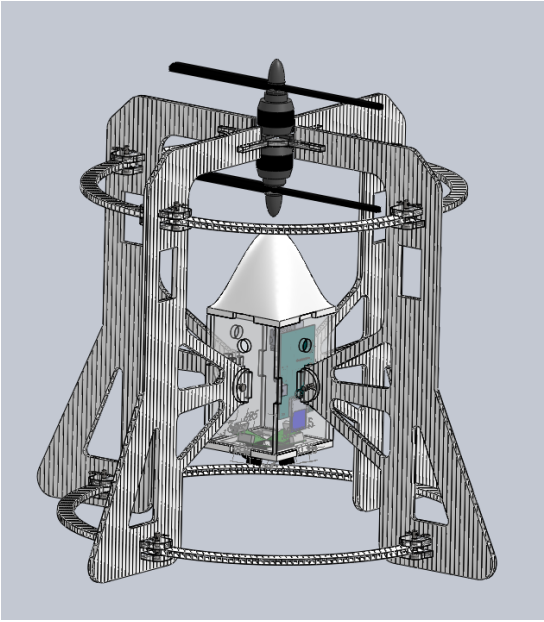


Figure 29: Updated outer frame with added propeller protection

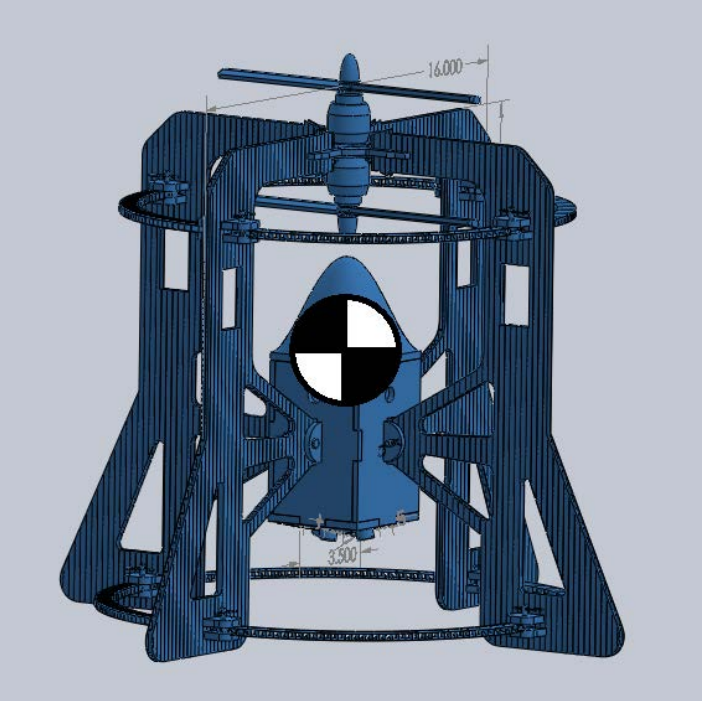


Figure 30: The center of mass of the system demonstrated by the CM symbol (units in inches)

After conducting launch tests with this prototype it was determined that the IPASS was too heavy to lift off using the brushless motors that were being tested. This chassis prototype weighed a little over 2.4 Kg, and the thrust generated by the Turnigy motors was below that. It was also observed that the kinetic energy from the propellers did not generate adequate torque to rotate the IPASS due to the large surface area of each quarter piece. The moment of inertia was too large due to the mass of the quarter pieces having a relatively large radius from the center axis of the chassis. This is demonstrated in Equation 3*: Moment of Inertia*.

$$\boldsymbol{I = \sum_{i=1}^{N} m_i r_i^2}$$

Equation 3: Moment of Inertia

Where m_i is the point mass a distance r_i from the axis of rotation. It is clear that the moment of inertia depends largely on the radius from the point mass to the center of rotation due to the fact that the radius is squared. This concern was addressed in the next iteration of the frame design by pulling the quarter frame piece closer to the central axis.

Iteration Three

The next design was a departure from the previous design which was based on the original prototype. At this point it was determined that the IPASS needed a slightly wider electronics box to fit all of the components. The frame quarter pieces were designed to be much smaller and were mounted directly to the sides of the electronics box using clevis pins and Delrin tabs. This allows for a much lighter, thinner chassis which is more conducive to rotation about its central axis while airborne. The thinner chassis can be seen in figure 36.

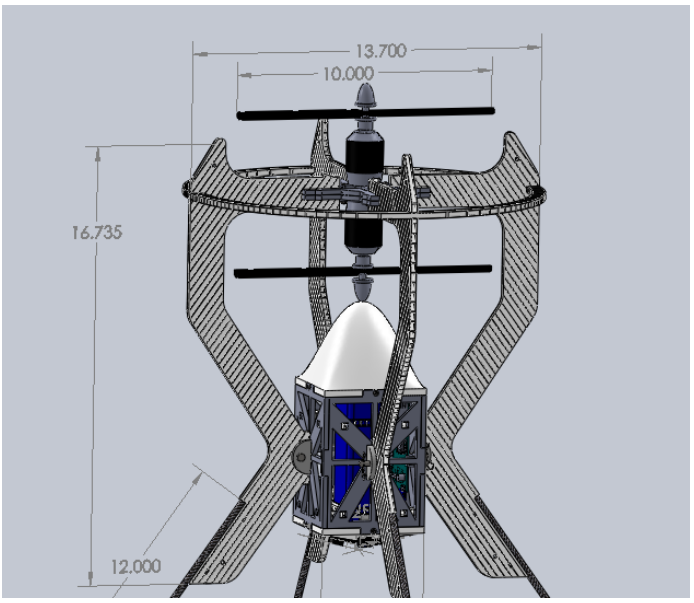


Figure 31: Design of the IPASS for a smaller moment of inertia (units in inches)

This design however does not use a spring-like attachment to the electronics box which was utilized in the previous design. The new prototype used the spring of the lower legs as shock absorbers. If there is enough force, however, the bottom of the electronics box may hit the ground, thus it was required to increase the distance of the electronics box from the ground.

In the next iteration, four carbon fiber tubes were attached to the frame quarters which absorbed shock. These tubes were one foot long each and attached to the chassis quarters using Zip Ties. These tubes absorb the impact of the fall and direct it around the core components. The part of the Coroplast frame quarter that is held flush to the Delrin sides of the electronics box will expand on the lower side and compress on the upper side as seen in figure 37. This flexion absorbs the impact of the collision so that the internal components are protected. This design was partially influenced by the Honeywell T-Hawk, shown in figure 38, which uses large rounded landing gear designed to absorb shock.

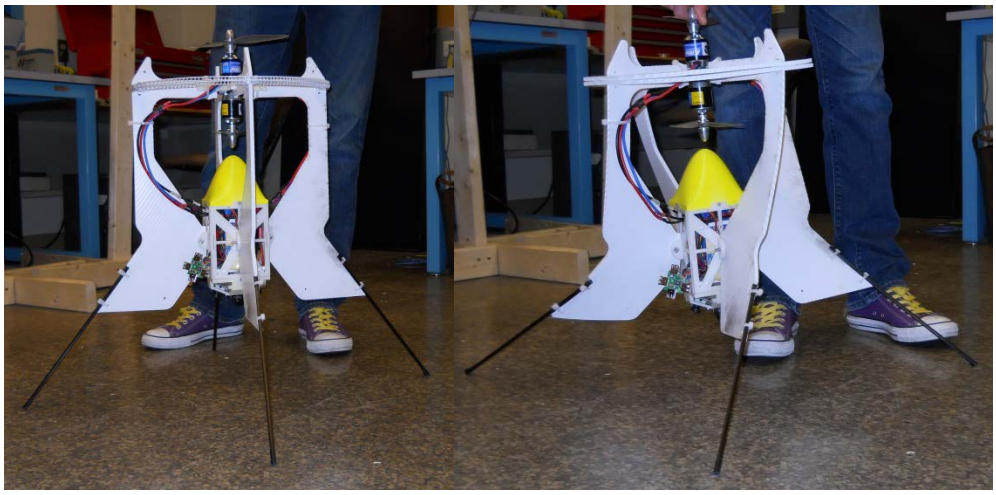


Figure 32: Carbon fiber springs to assist in shock absorption

Figure 33: Honeywell T-Hawk

After some initial testing including drops from a few feet it became apparent that the quarter frames needed to be stronger. To adress this, the pieces were cut so that the corrugation was more favorable to inhibit flexion. This is due to the area of the cross section where the flex occurs as can be seen in figure 39. Below, the image on the left will have more area in its cross section, which makes it resist flex.

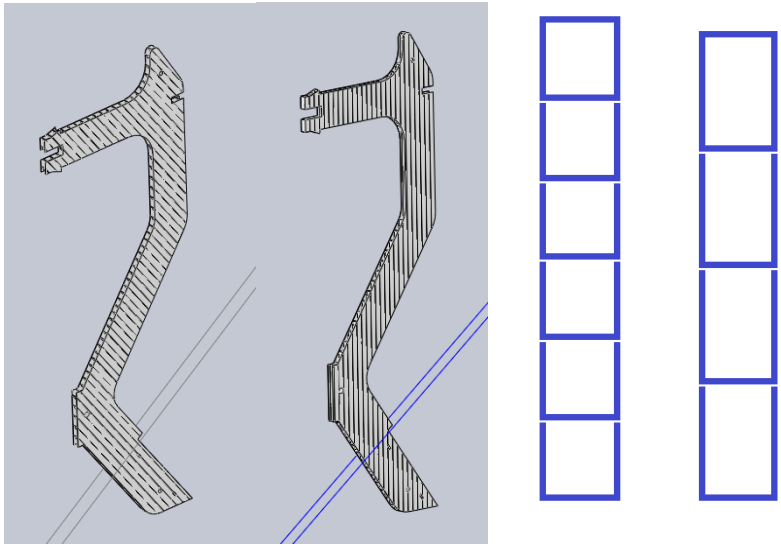


Figure 34: Comparison of differences in cross sectional area

This version of the quarter frame was designed in such a way that the surface area of the chassis when viewed from the side above the center of mass and below the center of mass is approximately equal as shown in figure 40. This allows the system to be less prone to rotation caused by external forces. If the IPASS encounters wind, the chassis will be more likely to translate rather than rotate, which is shown in figure 41.

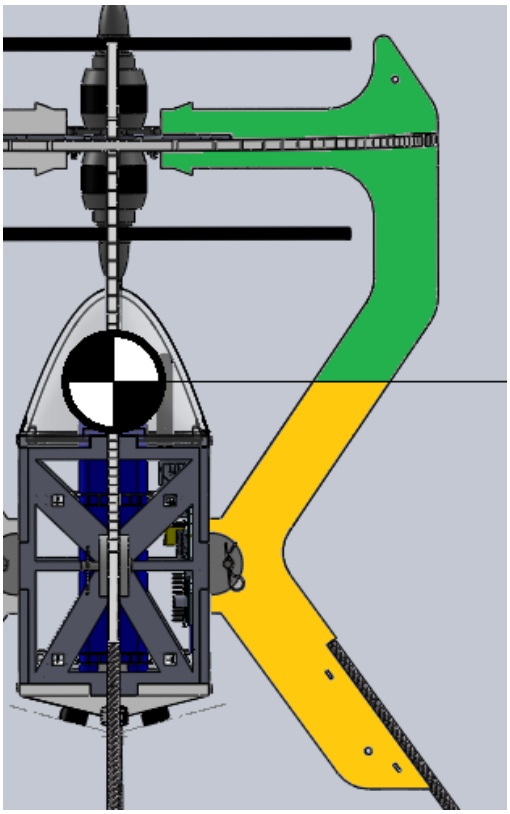


Figure 35: Surface area above and below the center of mass

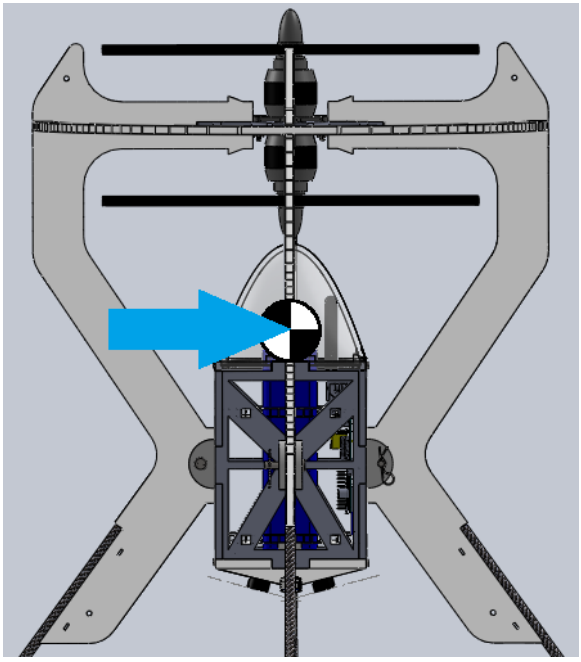


Figure 36: Force applied by wind

The formula for finding the force caused by wind is defined in Equation 4 where A is defined as the cross sectional area of the item, P is the wind pressure, and Cd is the coefficient of drag,

which is defined as 2 for flat plates, such as the flat quarter frames of the chassis or the sides of the electronics box [18].

$$Force = A * P * C_d$$

Equation 4: Force caused by wind

P is the wind pressure and is defined in Equation 5: Wind pressure. V indicates the wind speed in miles per hour.

$$Wind\ pressure\ (Psf) = .00256 * V^2$$

Equation 5: Wind pressure

After observing the design during testing, it became apparent that the propellers were vulnerable to damage in the event of improper landing. To address this issue a propeller protection ring was added to the design. This ring had slots that would mate perpendicularly with slots on the quarter frames to protect the propellers from colliding with the ground. This protective ring can be seen in figure 42 and figure 43.

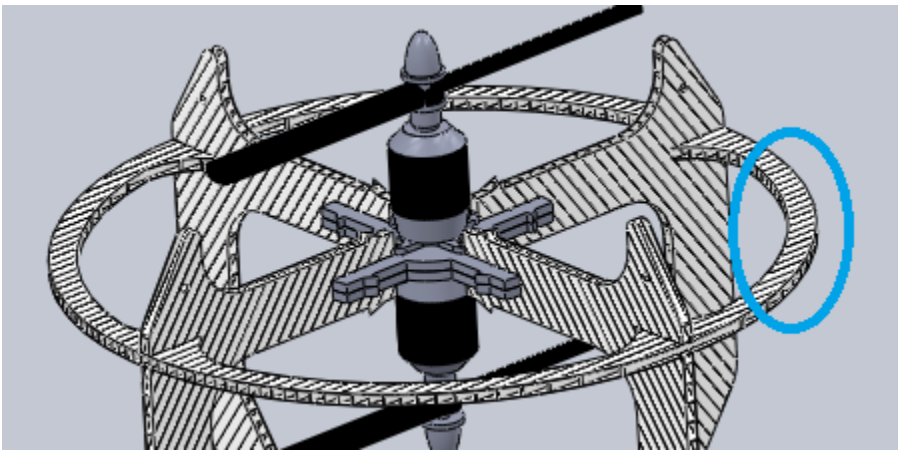


Figure 37: Propeller protection ring (circled)

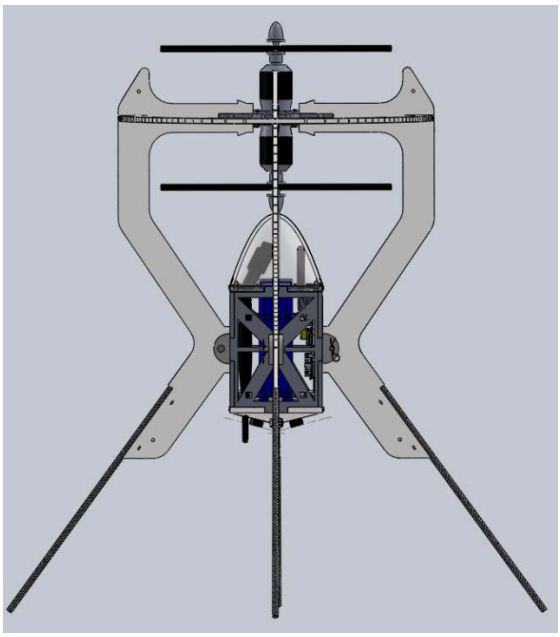


Figure 38: Third iteration of IPASS chassis design

Iteration Four

After attempting multiple launch tests it was discovered that stabilizing fins were required to allow the system to behave more like a propeller powered rocket. With this in mind, the lower part of the frame quarters were expanded outward in an attempt to increase stability and straighten the trajectory. This also adds strength to the frame during an impact. This final design can be seen in figure 44.

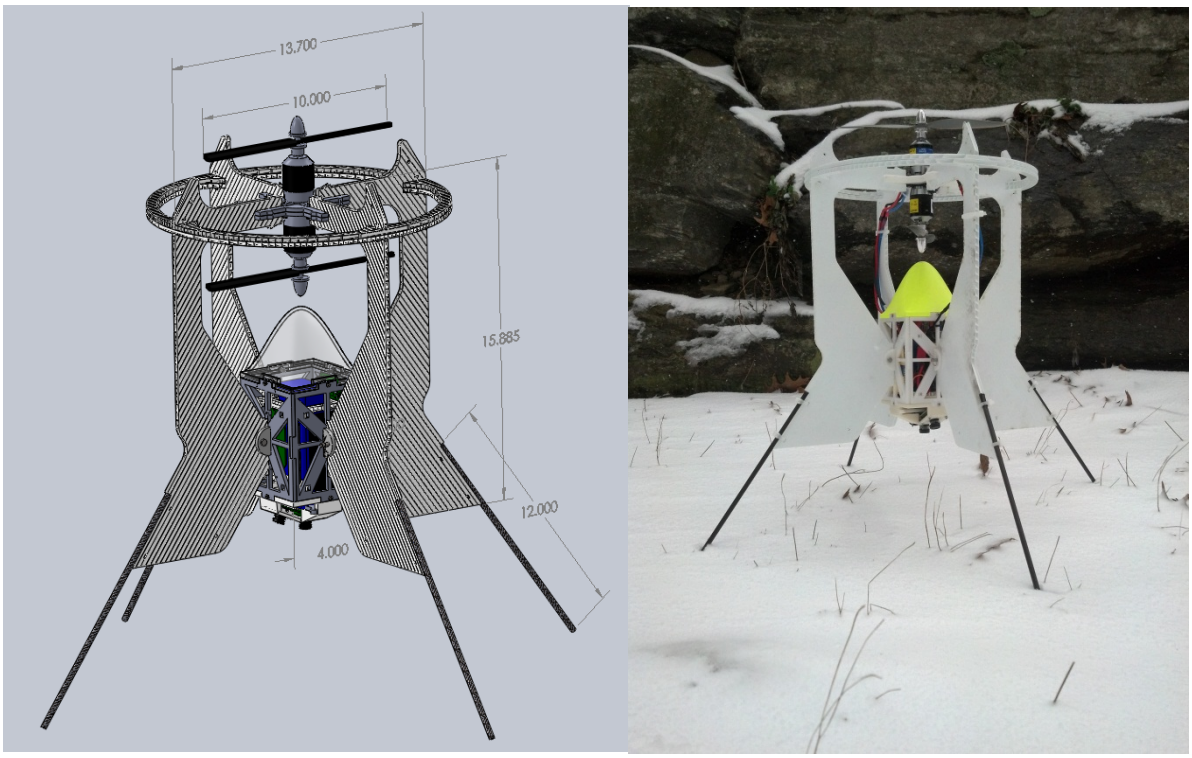


Figure 39: Final design and completed system (units in inches)

2.3.2 Electronics Box

The electronics box serves as a mounting solution and protection shell for the core electronics including embedded computing, sensors, cameras, and the power system. Originally the electronics box was sized to fit a standard size Li-Po battery along with a Raspberry Pi. The battery was intended to be stored inside the electronics box along with the embedded computing, the ESCs, sensors, and the cameras. The first iteration was made of solid laser cut acrylic with holes for wire routing to the motors. At this stage of the design a Raspberry Pi fulfilled the embedded computing requirement. The Raspberry Pi does not have any means of mounting, thus clips were designed to be 3D printed to secure the Raspberry Pi to the side of the electronics box. The original electronics box design can be seen in figure 45 and the mounting method can be seen in figure 46.

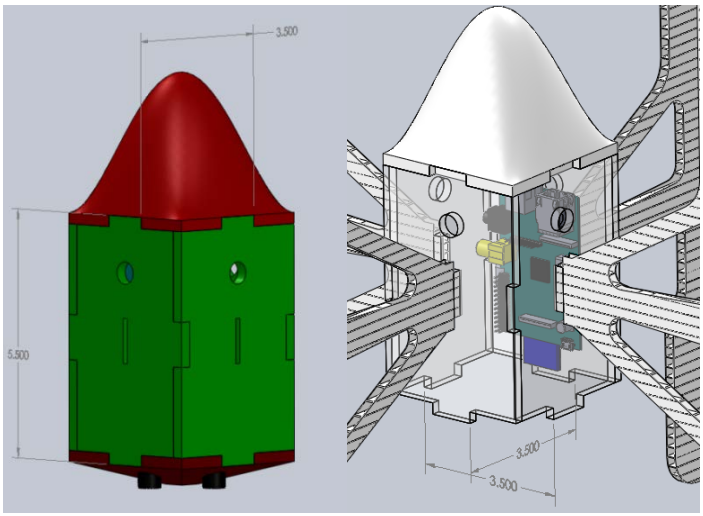


Figure 40: Original electronics box prototype (units in inches)

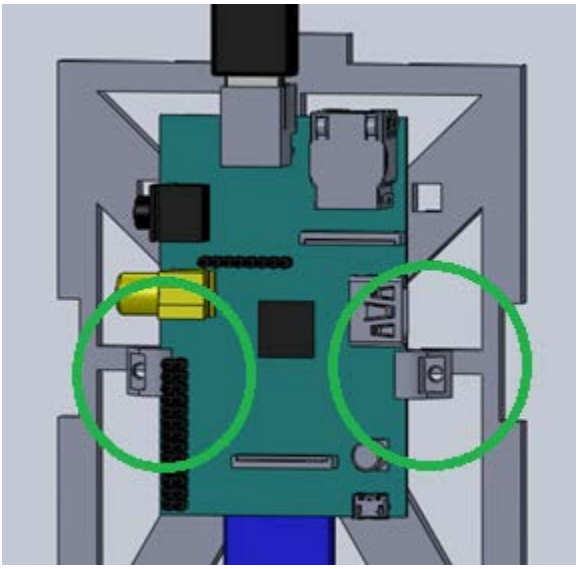


Figure 41: Original mounting for the Raspberry Pi (circled)

Electronics Box Material

The electronics box needed to be manufactured from a very durable material in order to protect the internal components. The overall system must be low cost so a light, inexpensive, and durable material was required. It would be possible to 3D print the electronics box but this would not be cost effective given the high price of printing large parts. Since the team had access to a laser cutter, plastics that could be laser cut were researched. Such materials included acrylic and Delrin.

Delrin was chosen for the electronic box due to its toughness. Delrin has a flexural modulus of around 8 GPa where acrylic has a flexural modulus of around 3 GPa [27, 28] . This makes the material less brittle than acrylic and tough enough to flex on impact in order to absorb energy from impact without breaking. Delrin of a 3/16 inch thickness was chosen due to its strength and ability to be laser cut. Thicker Delrin will deform when laser cut as its increased

thickness requires additional heat. If the Delrin is too thin then it will not provide adequate protection. The sides of the electronics box have triangles cut out to reduce weight, include mounting holes for the 3D printed parts, and chassis mounting tabs. The entire electronics box uses two pairs of interlocking tabs and slots on each side. These Delrin sides are shown in figure 47. This reduces the total number of different parts required. These holes will be covered to reduce drag using a plastic film or adhesive backed tape.

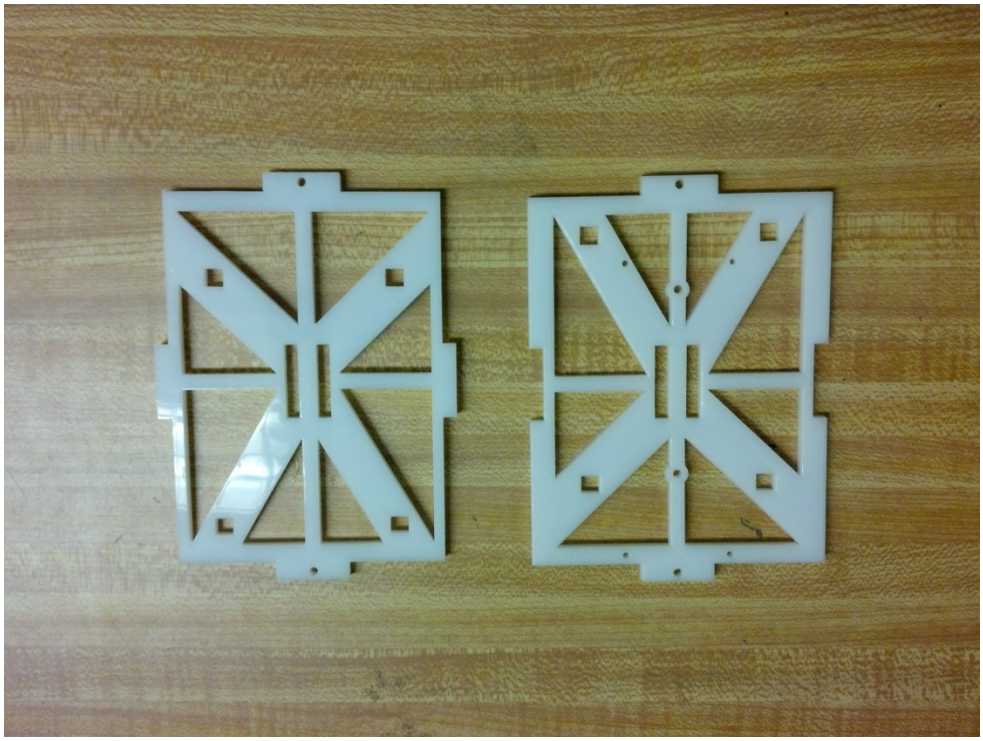


Figure 42: Delrin electronics box sides

The team knew that the battery was going to be the heaviest component in the system and therefore it was mounted in the center of the electronics box for the purpose of aligning the center of mass as seen in figure 48. The battery is held using two Coroplast supports. These supports also work as shock absorbing cushions should the IPASS land on its side. The bottom of the battery is supported by the camera mount assembly which also protects the cameras from being damaged in the event of a downward crash. The Coroplast supports also have recessions for clearing the electronic components, wire routing, and mounting the Wi-Fi antennas.

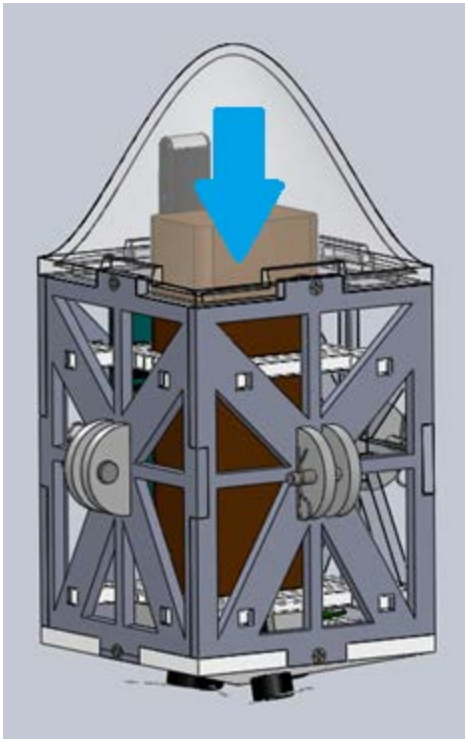


Figure 43: Battery location (indicated with arrow)

The electronics have been mounted on the inside surfaces of the four sides of the electronics box. On one side, the Jameco circuit prototyping board, as seen in figure 49, which holds the microcontroller that controls the motors, is mounted. This board is mounted using standoffs and screws for a secure and durable mount. On the same side is the radio receiver used to read signals from the radio controller. Opposite of this side is the main computing for the IPASS, which is wired to the cameras, as well as the motor controlling computer.

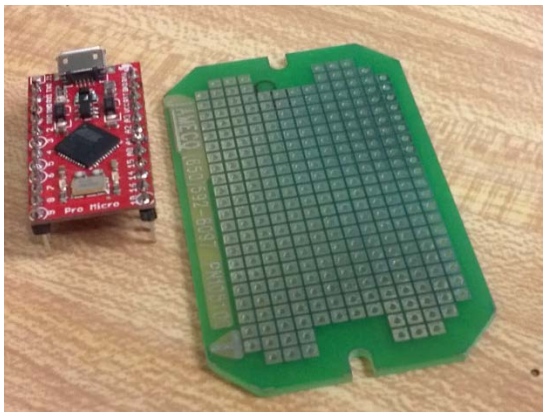


Figure 44: Jameco prototyping board

After switching to the Gumstix computer with the Tobi breakout board, the design was updated to mount the board directly to the side of the electronics box using standoffs and screws. On the remaining two sides are the electronic speed controllers, used for transferring power from the

battery to the motors as well as control signals from the motor controller. This setup can be seen in figure 50.

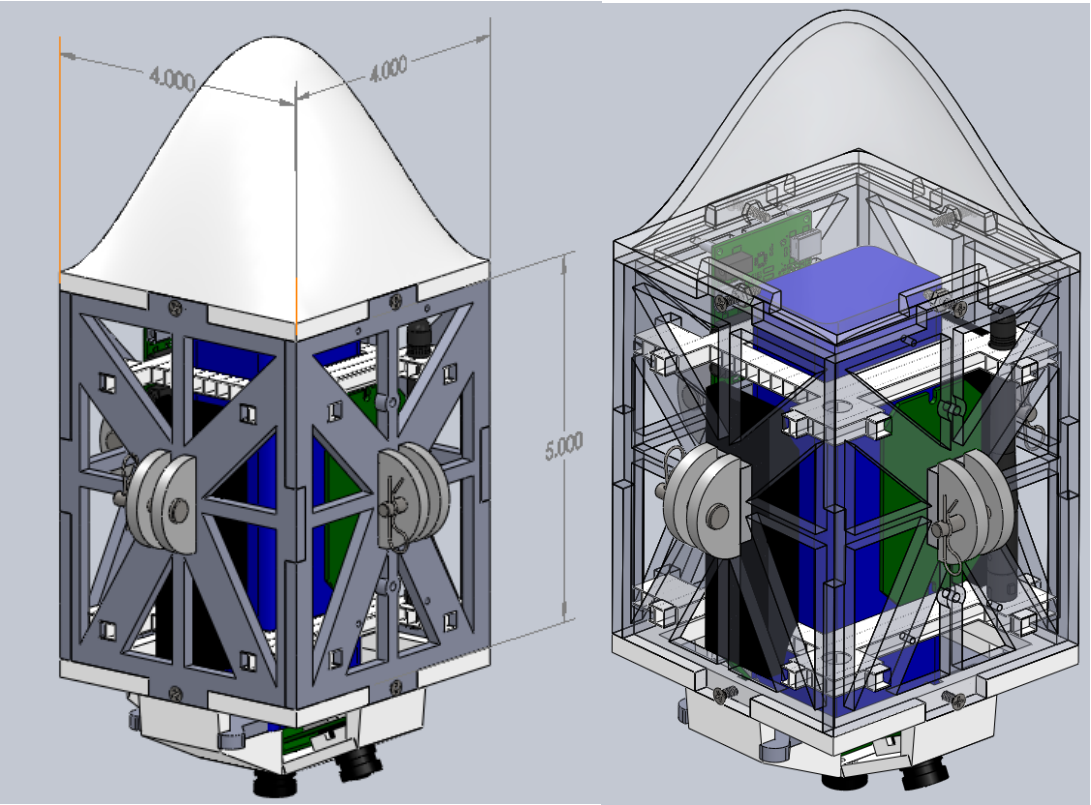


Figure 45: Final electronics box component layout (units in inches)

Top Cone

On top of the electronics box sits the aerodynamic top cone. This cone is designed to guide the flow of air from the propellers around the electronics box decreasing parasitic drag induced by the cross section of the electronics box. The force due to drag is defined in Equation 6: Drag equation.

$$F_d = C_d * \frac{1}{2} * \rho * V^2 * A$$

Equation 6: Drag equation

Where C_d is the drag coefficient, ρ is the density of the fluid (air), V is the flow velocity, and A is the characteristic front area of the body. The drag coefficient can be approximated by comparing the shapes from known coefficients to the shape of the nose cone. It is known that a hollow half sphere has a drag coefficient of around 0.38 and a streamlined body has a drag coefficient of around 0.04 [18].The design of the nose cone is an intermediary between these two shapes, but much closer to the hollow half sphere, so the drag coefficient is approximated to be 0.25.

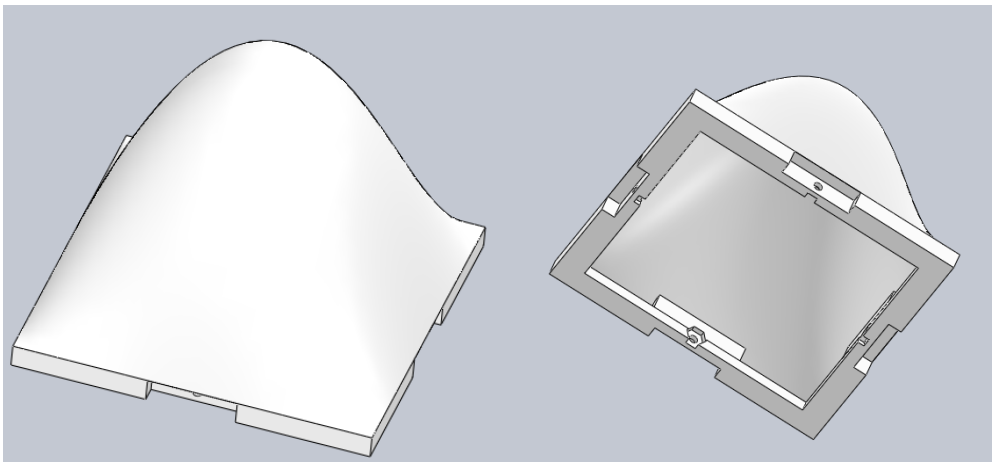


Figure 46: Top cone of the electronics box

This part, seen in figure 51, was made using WPI's Dimension 3D printer which extrudes Acrylonitrile Butadiene Styrene (ABS) plastic in layers to create a part. It has three density settings that change how the inside of the part is filled with plastic. For this component, the lowest density setting was chosen to reduce weight. On each mating point between the cone and the Delrin sides there is a recessed hole designed to fit a 4-40 hexagonal nut which will prevent the rotation of the nut during assembly.

Camera Mount

On the bottom of the electronics box is the camera mount as seen in Figure 52. The camera mount is designed to hold four cameras in a fixed position as well as to prevent the battery from damaging the cameras during landing. The original design required the use of four cameras. The mount was designed to hold the cameras in a pyramid formation with overlapping fields of view shown in figure 52. Additional view of the camera mount can be seen in figure 53 and figure 54. This pyramidal design allowed for a wide field of view which was intended to provide usable image data. On the downside, however, there was significant overlap and therefore redundant image data.

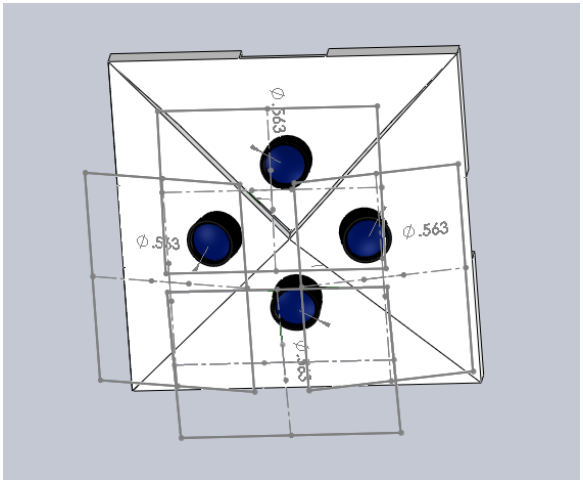


Figure 47: Four camera layout design

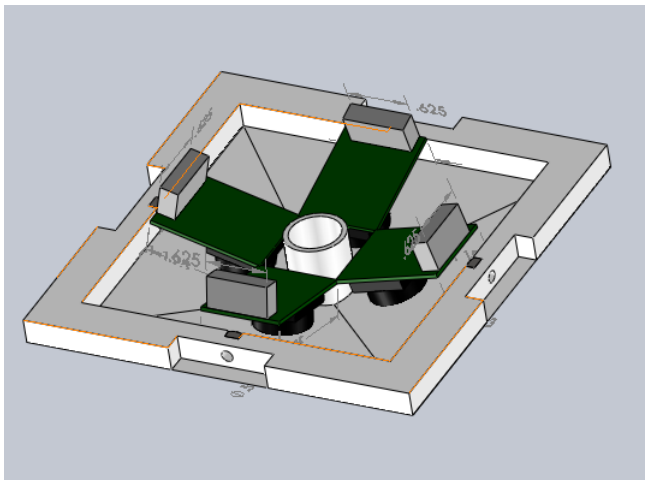


Figure 48: Camera mounting viewed internally

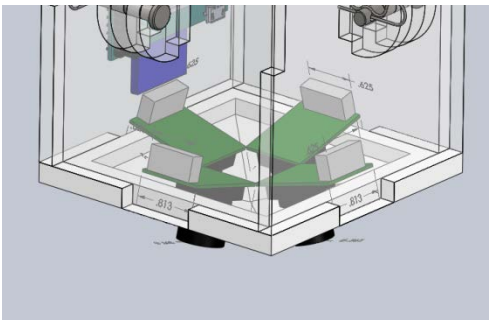


Figure 49: Initial camera mount attached to electronics box

Due to a change in camera model during design, the camera mount was modified to hold three cameras. This design can be seen in figure 55 and figure 56. These cameras were slightly larger than the previous cameras. These cameras are mounted in a way such that the images are stitched three in a row to make a wider field of view. This part uses three raised camera lens mount holders that tightly hold the camera lenses and are designed to mitigate motion. This design uses a separate Delrin plate to protect the cameras from the battery as can be seen in figure 57.

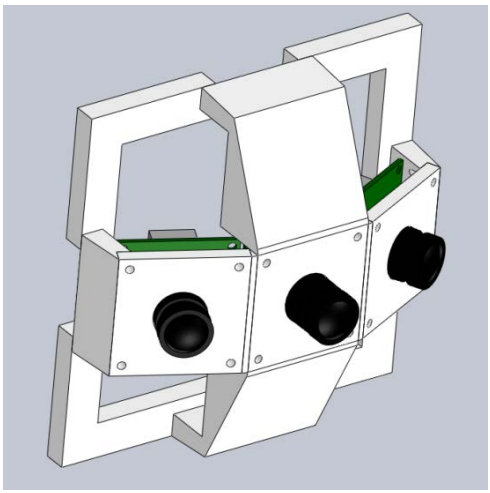


Figure 50: Three-camera mount design viewed externally

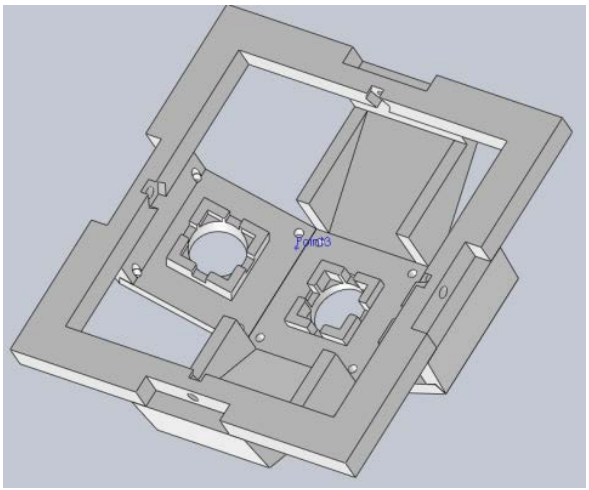


Figure 51: Three-camera mount design viewed internally

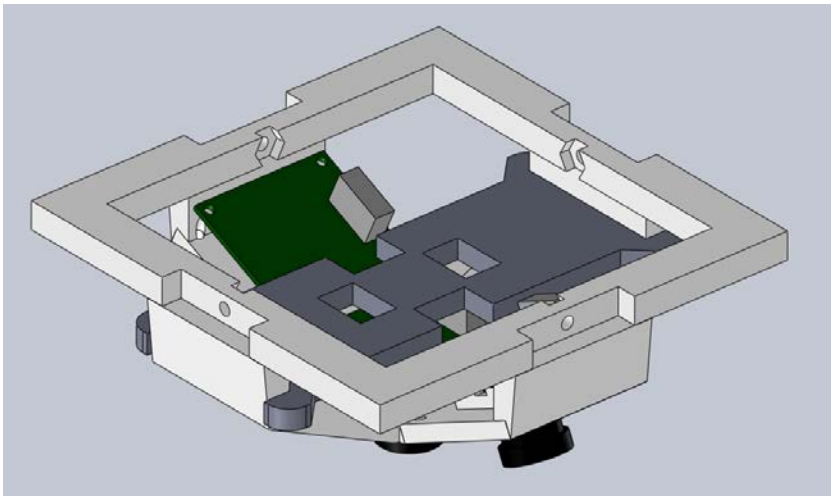


Figure 52: Three-camera mount with Delrin protection piece

The electronics box is attached to the quarter frames by Delrin tabs and clevis pins. The Delrin tabs are press fit into the electronics box. The tabs lock the chassis quarters in place using 3/16 inch clevis pins held in place with cotter pins as can be seen in figure 58. By using clevis pins and tie wraps, the IPASS can be broken down and stored as well as assembled in the field without tools.

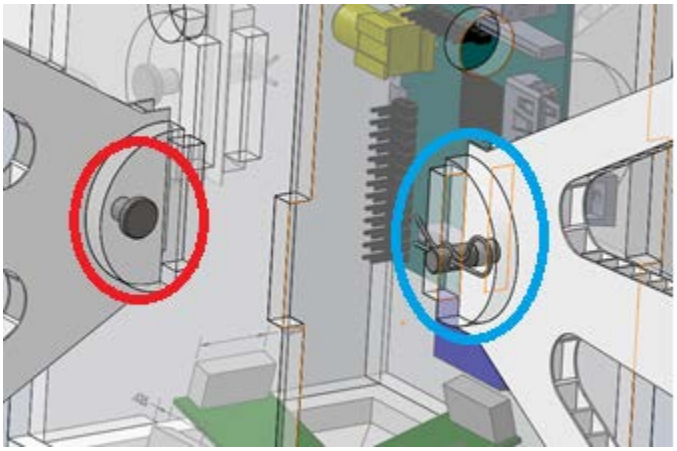


Figure 53: Quarter frame mounting solution. The red circle highlights a clevis pin. The blue circle highlights a Delrin tab.

2.4 Power System

Table 5 details the power requirements for the components intended for use in the final IPASS design.

Table 5: Power requirements of the IPASS components

Component	Current (mA)	Voltage (V)
Gumstix	1000	5
Arduino	200	5
Cameras	500	5
GPS	<50	5
Motors	50000	12

To power the IPASS, a Lithium Iron Phosphate (LiFePo) battery was chosen. LiFePo batteries feature similar power ratings to a Li-Po battery but are more stable. This makes them ideal for aerial systems with high power requirements. The ESCs connect directly to the LiFePo battery, supplying the motors with the required 12V. The 5V BEC from the ESCs provide power to the embedded computing and sensor systems.

2.5 Embedded Computing

To satisfy the IPASS requirements the on-board embedded computing system needs to be capable of the following:

1. Transfer data wirelessly
2. Interface with multiple devices
3. Be small and lightweight

The team explored three embedded computing systems for use on the IPASS: The Raspberry Pi and Arduino Pro Micro were originally chosen. A Gumstix Overo FE COM later replaced the Raspberry Pi in the design process.

2.5.1 Consideration: Arduino Pro Micro

The IPASS has two Pulse Period Modulation (PPM) controlled brushless DC motors. The PPM protocol is a very time intensive task and the team decided that a smaller secondary processor would be responsible for controlling the motors. This processor is also responsible for receiving pose data of the IPASS. The Arduino Pro Micro was chosen due to its low cost, small size, and the team's familiarity with the Arduino environment. The Arduino Pro micro includes four channels of 10-bit ADC, five Pulse Width Modulation pins, twelve DIOs, and hardware serial connections Rx and Tx. The Pro Micro also has a voltage regulator, allowing it to accept voltages up to 12V. Although the team chose to use an Arduino Pro Micro, any Arduino model would be capable of fulfilling this requirement due to the capabilities of all Arduino models.

2.5.2 Consideration: Raspberry Pi

The team initially chose the Raspberry Pi due to its low cost, ease of integration, high processing speed and market popularity. The Raspberry Pi contains a 700 MHz Advanced RISC Machines (ARM) 11 processor and is capable of connecting to other devices via Ethernet, USB, I^2C, SPI and UART. The Raspberry Pi runs a full Linux environment booted from an SD card. The Raspberry Pi model B is inexpensive and costs $35.

Since the IPASS needs to transfer data wirelessly, a USB Wi-Fi device was obtained on recommendation of compatibility with the Raspberry Pi. This allows the IPASS to communicate over the 802.11 wireless standard. The IPASS is also required to gather information on its orientation and position using external sensors. Initial designs include a Global Positioning System (GPS) receiver and an Inertial Measurement Unit (IMU). The GPS receiver was connected to the Raspberry Pi via USB and the IMU was connected to the Arduino Pro Micro to be accessed via the UART.

Since the Raspberry Pi runs in a Linux environment, development was simplified as the team was able to design programs in ANSI C rather than in a specialized microcontroller environment. The Linux environment made the peripherals more accessible as drivers were already developed by the Raspberry Pi Foundation.

During development, issues were identified relating to the Serial Peripheral Interface (SPI) communication with the Raspberry Pi in which the first bit of a transmission was consistently dropped. The initial cameras communicated with the Raspberry Pi through SPI. Despite functional cameras, these complications resulted in unusable incoming SPI data to the Raspberry Pi from the cameras. In order to solve this problem the team investigated additional computing solutions.

2.5.3 Consideration: Gumstix

The team purchased a Gumstix Overo FE COM for the embedded computing to replace the Raspberry Pi based on its better documentation and more robust computing system. The Gumstix, like the Raspberry Pi, has a native Linux environment and the ANSI C code originally developed could be readily ported to the Gumstix.

The Gumstix has a powerful processor for an embedded computer but has a limited selection of peripherals including a Wi-Fi receiver, Bluetooth receiver, camera connector for proprietary cameras, and two 70 pin AVX connectors . Each Gumstix model can be mounted onto an expansion board featuring a variety of peripherals.

IPASS design requirements call for wireless data transmission in a variety of environments. Testing was conducted in Worcester, Massachusetts where temperatures typically drop below freezing. The Gumstix Overo FE COM was purchased because it is the only Gumstix model capable of operating in freezing conditions. This particular model had an on board Wi-Fi card and comes with one antenna for 802.11 and one for Bluetooth.

The team purchased two different development boards for the Gumstix. The Alcatraz, as seen in figure 59, is a full break-out board giving access to all 140 I/O pins on the Gumstix. This would give the team the most developmental capabilities. However, with a size of 4 x 5 in., it was decided that the Alcatraz was too large to be integrated within the IPASS given the small size of the electronics box.

Figure 54: Alcatraz development board [19]

The Tobi expansion board features 100 fewer breakout pins, one USB port, one HDMI port, one Ethernet port, and is 4 1/8” x 1 ½”. The Tobi board was chosen because it still fulfilled all IPASS design requirements while having a small form factor as can be seen in figure 60. Later in the development process, a USB hub was integrated to facilitate connection with additional sensors.

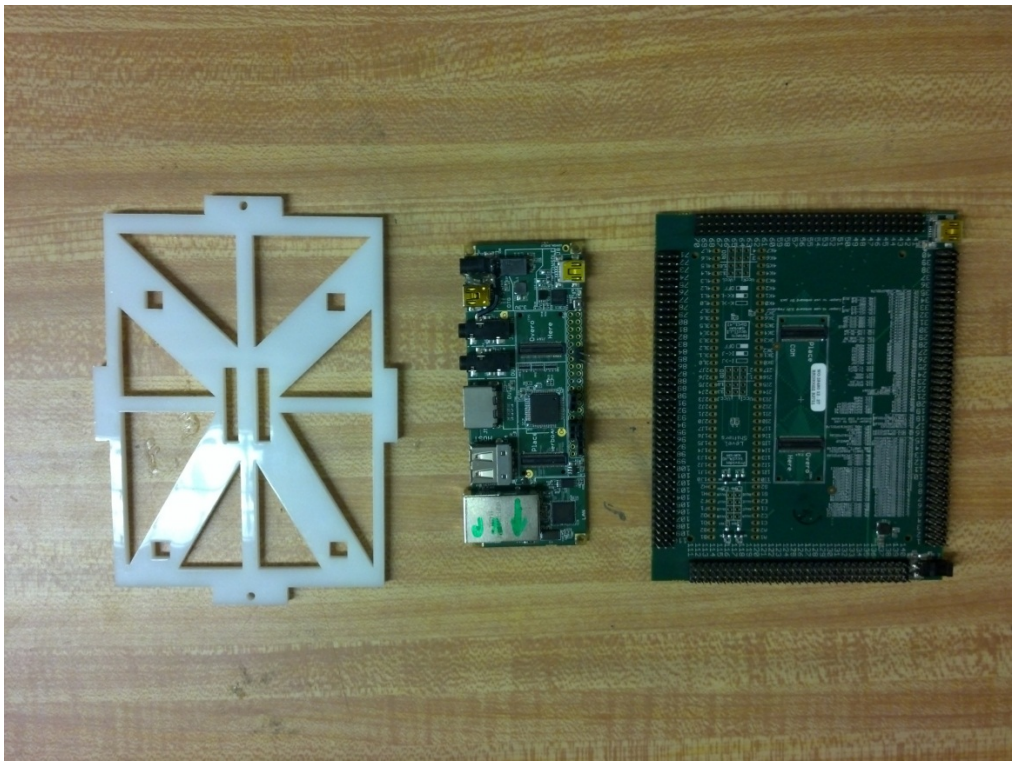


Figure 55: Electronics box Delrin panel, Tobi board, and Alcatraz size comparison.

2.5.4 Inter-processor Communication

The IPASS was initially designed with the intention of using a Raspberry Pi as the primary embedded computing system and was required to communicate with an Arduino Pro Micro. A Gumstix Overo was selected for use in the final design of the IPASS to communicate with the Arduino.

Consideration: Raspberry Pi to Arduino

Because of the team's choice to have multiple processors on the IPASS, a communications medium was developed. A serial connection via UARTs of the Raspberry Pi and Arudino with a maximum transfer speed of 115.2 Kb/s was chosen due to its simplicity. This would allow the Raspberry Pi to request position and orientation data from the Arduino.

Implementation: Gumstix to Arduino

The Tobi board gives access to 40 GPIO pins on the Gumstix. These pins are used to send data to the Arduino Pro Micro. During this stage of the design, the Gumstix did not need to receive information from the Arduino. The Gumstix only needs to relay three commands to the Arduino: Launch, Land, and Abort. These are discussed in depth in 2.9 Software.

2.6 Vision System

When choosing cameras for the IPASS there was several design requirements to be taken into consideration. Cameras are needed that are small and lightweight to ensure a light load on the IPASS. Because images are not being streamed rapidly to the ground station, a high fps is not necessary. The power consumption of the cameras is not a design concern. The design calls for a multiple camera system therefore the cost of each sensor needs to be minimized.

The team determined useable image data as images in which a human can easily be identified at 100 ft. A test was conducted to determine the individual image resolution required of each camera. A team member was photographed at different resolutions and varying distances using several cameras the team had available. A comparison of some of these images can be seen in figure 61 while full details of this test are documented in Appendix C. The team determined that a human at a distance of 100ft from the camera could be easily distinguished in a 640 by 480 pixel image. Therefore, only cameras capable of this resolution were considered.

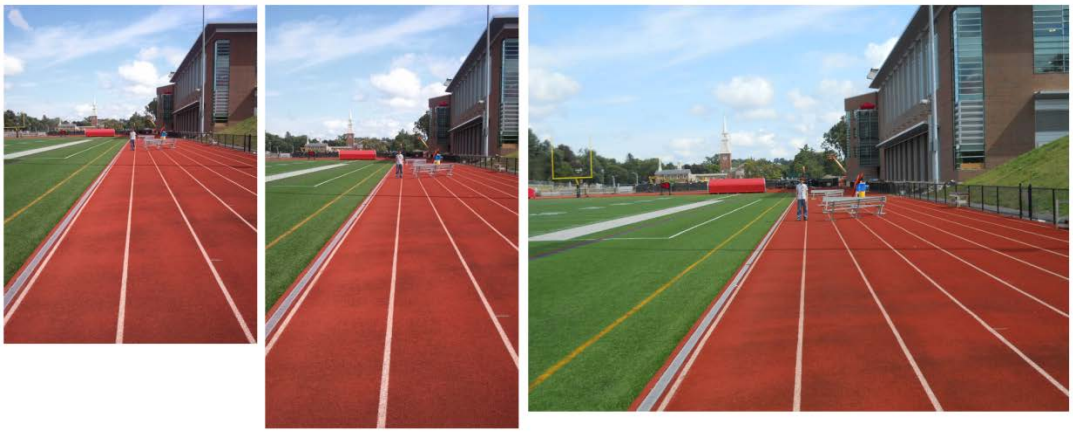


Figure 56: Images of a human at 100ft. From left to right, the resolutions are 640 by 480, 800 by 480, and 1024 by 768 pixels. These images have been scaled down.

The speed at which cameras could take and send images was an important attribute. The IPASS needed to be capable of taking and sending multiple images to the embedded system during operation. Cameras with only a serial connection were not investigated because serial connections were determined to be too slow. SPI and USB connections were determined to be the most feasible due to their speed and ease of use. Table 6: IPASS Camera Pugh Chart shows a Pugh chart created to help analyze camera options.

Table 6: IPASS Camera Pugh Chart

Camera		C329 board	CMOS Camera	SPI High Speed JPEG Camera	USB Cameras
Criteria	Weight				
Size and Weight	1	5	5	5	4
Max Res	2	5	5	5	5
Max FPS	1	5	5	5	5
Ease of Interface	3	3	3	3	4
Speed of Interface	4	4	2	4	4
Availability	3	5	5	5	5
Power	1	5	5	5	5
Cost	2	2	5	3	4
Total		69	67	71	75

The sensor used for the imaging of the camera affects how quickly the camera are able to capture a picture. The two major sensor types are CCD and CMOS. CCD sensors take a picture by exposing all imaging pixels at once to capture image data. CMOS sensors take a picture by exposing sets of imaging pixels rapidly. CCD is marginally preferable for use on the IPASS because they take a picture faster than CMOS sensors.

The ground sampling distance (GSD) of the cameras also needs to be considered. The GSD of a system relates to how much real-world space one pixel of a sensor captures in a single image. The GSD determines how many pixels in a picture will represent an object. Equation 7 was used to calculate the GSD.

$$GSD = pixel\ element\ size * height\ above\ ground / focal\ length$$

Equation 7: GSD calculation

GSD is important for all aerial imaging applications. For example, an unacceptable GSD at 100 feet would result in a human appearing indistinguishable in the resulting image. The focal length is the distance from the lens to where the image converges and us the easiest factor to change in the GSD calculation. The pixel element size would change based off of camera choice.

Assuming an average human is 70 in. tall and 35 in. wide, a 2 in. GSD at 100 ft. would represent this human in 595 pixels. In Figure 61 above, the human in the 640 by 480 pixel image is represented by a total 525 pixels. Therefore, an acceptable GSD for the IPASS would be about 2 in.

2.6.1 Consideration: C329 Camera Module

The first camera module selected by the team was the C329 board as seen in figure 62. This was a CMOS sensor mounted to a circuit board interfaced through SPI. This model was chosen over USB because SPI communications would be faster. In addition to the typical power, ground, clock, MOSI, MISO, clock select, and slave select pins associated with SPI, the C329 SPI connection also featured a hold pin. This pin acts as a ready pin to notify the computer when the camera is able to receive commands. Because the CMOS sensor did not come with a lens, a 3.6mm lens was purchased with this module. This lens resulted in a GSD of 1.93 in. at 100 ft.

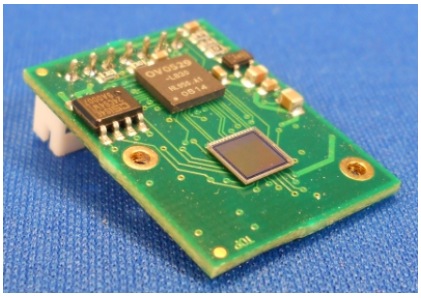


Figure 57: The C329 camera module [21]

Four C329 modules would be used on a single IPASS to provide a significantly large image of the ground. At 100ft, each camera would be imaging 100 by 75 ft. of ground. Each camera would be mounted so that there was overlap in images to allow for a higher quality

stitched image. For a detailed description about the C329 camera mounting design, please see 2.3.2 Electronics Box.

The C329 is able to send and receive commands successfully with an Arduino Mega. This established that the SPI communication between the camera module and an embedded system were functional. The C329 was then moved to be tested on the Raspberry Pi. It was found that the Raspberry Pi was capable of sending commands to the C329 during testing, but was unable to receive useful data. One bit of data was consistently dropped resulting in unusable data. The team verified this problem as a system malfunction by observing the C329 and Raspberry Pi communication on an oscilloscope. The C329 board was confirmed to be sending accurate commands but the Raspberry Pi was not reading in the data correctly. After further research the team found that it is a common occurrence for the SPI line to malfunction on a Raspberry Pi. The team reevaluated the current design and selected a new embedded computing system and cameras due to these errors in communication.

No images were ever successfully taken with the C329 module. During development, the team contacted the distributer and found that the specification sheet provided online was inaccurate and requested example code. The example code provided was designed for a specific subset of microprocessors not associated with the Raspberry Pi and deemed not useful.

2.6.2 Consideration: Caspa FS Camera

Due to the faults associated with SPI communication between the C329 and Raspberry Pi, alternative camera options were explored in addition to a new embedded computing system for the IPASS. The Gumstix was chosen not only because of its computational advantages over the Raspberry Pi, but because there is a camera module designed specifically to be interfaced with the Gumstix. The team purchased a full-spectrum Caspa camera, as seen in figure 63, to use on the Gumstix. The camera module comes with a 3.6mm lens and has a GSD of 1.93 in. at 100 feet.

Figure 58: The Caspa camera module [22]

The Caspa interfaces with the Gumstix through a proprietary 27-pin parallel connection. The video capturing program MPlayer was used to take video and images with the Caspa though the Linux terminal. Images were then sent via Secure Copy Protocol (SCP) to the team's storage server where they could be viewed. A detailed description on how to take pictures with MPlayer

can be found on the Gumstix wiki. To stream video from the Gumstix, the following command can be used:

```
# mplayer tv:// -tv driver=v4l2:device=/dev/video0
```

To take a picture, the mplayer command is called with different modifiers. An example of how to take a picture and send it is shown below. The image called “0000001.jpg” would be sent to the user “ipass” at IP address “10.42.43.2” and put into the home directory.

```
# mplayer tv:// -vo jpg -ss 1 -frames 1 -loop 1 –tv
driver=v4l2:device=/dev/video0
# scp 00000001.jpg ipass@10.42.43.2:/home
```

Pictures were taken with the Caspa at about a 1.5 sec. rate. The team was also able to stream video data to a monitor connected to the Gumstix. All images and video captured with the Caspa had a resolution of 640 by 480 pixels. To focus the Caspa, a set-screw in the lens was loosened during a video stream. The lens was then manually focused until the picture on-screen became clear and the set-screw was put back in place.

The Caspa was significantly easier to interface than the C329 board. Instead of locating and initializing SPI communications, the Caspa featured a simple plug-in connection. However, only one Caspa can be connected to a Gumstix as each board only features one 27-pin connection. Because a single camera would not provide enough ground coverage, the Caspa camera was not chosen for the IPASS.

2.6.3 Implementation: SB101C USB CMOC Board

The SB101C camera module, as seen in figure 64, was obtained as the final camera solution for the IPASS. These modules have a GSD 1.93 in. at 100 feet. At 100ft., a single camera would image a 100ft. 75ft. plot of ground. These cameras are about 1.25in.^{3} in volume which makes them easy to integrate within the IPASS and are comparatively inexpensive to the previous camera solutions used. The USB interface also allows for easy integration with multiple systems.

Figure 59: The SB101C USB camera module [28]

Out-of-the-box, these cameras come with a 5 pin to USB-B connection. However, the team wanted to use USB-A connections which are more common and allow for easier expansion of the system. Custom USB-B to USB-A connections were made for these cameras during development. A four port USB hub was needed in conjunction with these cameras because the Gumstix Tobi expansion board only has one USB port. In system level testing, the SB101C camera connections were soldered directly to the USB hub to ensure a secure connection. Three SB101C modules were used on the IPASS. For a detailed description about the SB101C mounting, please see 2.3.2 Electronics Box.

There is no method of holding the SB101C lens in place other than with the threading on the lens. The SB101C was focused in the same way as the Caspa except that once in focus, the lens was hot-glued in place to prevent errant lens movement.

Images from the SB101C module are taken with the MPlayer program. Using a script, pictures can be taken from all three cameras using the MPlayer software and sent to the ground station via SCP. The time taken to save an image from the SB101B camera and the Caspa FS proved to be almost identical. This is because the actual transfer time of the image data itself is miniscule when compared to the time it takes to save the image to the Gumstix memory.

2.7 On-board Sensing

Onboard sensors are required in order to track the position of the IPASS during flight and recovery. The team explored two sensor options in order to track position: an IMU and GPS receiver.

2.7.1 Consideration: Global Positioning System Receiver

GPS technology was explored as a GPS receiver can track position and elevation with acceptable accuracy. GPS receivers manufactured by Globalsat were investigated due to their low cost and variety of specifications. The team chose to the use the ND-100S USB receiver for its ease of use and small size. The GPS receiver is connected to the Gumstix through the four port USB hub. While the GPS receiver was intended to operate in tandem with the camera system on the IPASS, difficulties in which the GPS receiver and cameras would draw too much power from the Gumstix led the receiver to not be included when field testing the IPASS

The factors taken into account when choosing a GPS receiver were price, size, and startup time. Price needed to be minimized in order to keep the overall cost of the IPASS low. Size needed to be considered in order to ensure that the electrical components fit within the electronics box. Startup time was considered as the IPASS needs to be quick to set up and deploy. A fast startup time will ensure that the IPASS can be deployed within a reasonable time frame. A comparison of the GPS receivers analyzed can be found in
Table 7. GPS receivers from the Globalsat company were analyzed due to the low cost of Globalsat's products and the ease of integration (all their receivers connected via USB). As many of their GPS receiver models had only small differences in their specifications, the smallest model was chosen.

Table 7: Comparison chart of GPS receiver selection

Criteria	Weight	GlobalSat ND-100S USB Dongle GPS Receiver	GlobalSat BU-353 Cable GPS with USB interface (SiRF Star III)	GlobalSat BU-353S4 Cable GPS with USB interface (SiRF Star IV)
Image				
Price ($)	1	2	1	3
Size (mm)	5	3	2	1
Hot Start (sec)	2	3	3	1
Warm Start (sec)	3	3	3	2
Cold Start (sec)	4	2	2	3
Total		40	34	28

Once the GPS receives a satellite connection, its position and elevation data can be written to a .txt file. This data is stored in the National Marine Electronics Association (NMEA) format which is capable of storing multiple types of data. These data types include bearing, geographic position, latitude, longitude, heading, altitude, and more. To meet IPASS design requirements only latitude, longitude, and elevation data were needed and thus were parsed out of the NMEA data using a Linux script. A second script then sends this data to the ground station where the data is displayed on the GUI.

In order to evaluate the capabilities and accuracy of the GPS unit the team performed three tests. These tests involved one team member moving the GPS receiver across the WPI football field while walking, jogging, and sprinting. The GPS data collected during these tests was then parsed, plotted on Google Maps, and then compared to the path actually followed. The results of these tests can be seen below in figure 65. These tests demonstrated that the GPS was accurate to six meters horizontally and half a meter vertically. Specific details of this test can be reviewed in Appendix F.

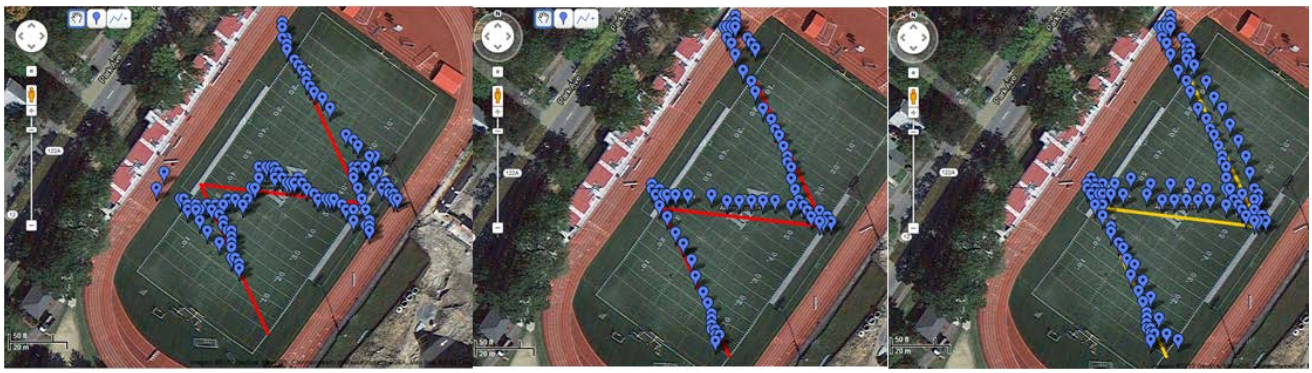


Figure 60: GPS receiver test data captured while team member was, from left to right, walking, jogging, and sprinting.

2.7.2 Consideration: Inertial Measurement Unit

The integration of an IMU is discussed here for completeness and future expansion. A 9-Degree Of Freedom (DOF) IMU was to be integrated in order to track the pose of the IPASS. IMUs vary greatly in price and performance levels and typically feature a linear relationship between price and performance. The team set a price limit of $100 for the IMU in order to maintain reliable data and low cost. The RoBoard RM-G146 was chosen because it was below the price limit and featured the best precision compared to other IMUs. While other options were available, the similarity in specifications made comparison between different models unnecessary.

The RoBoard IMU communicates via I^2C protocols. Arduino code was developed to capture all nine data values from the three sensors with one command. IMUs typically display noisy data over long periods of operation. To resolve the noisy and potentially inaccurate data a running average filter was implemented in code. The effects of this filter can be seen in figure 66.

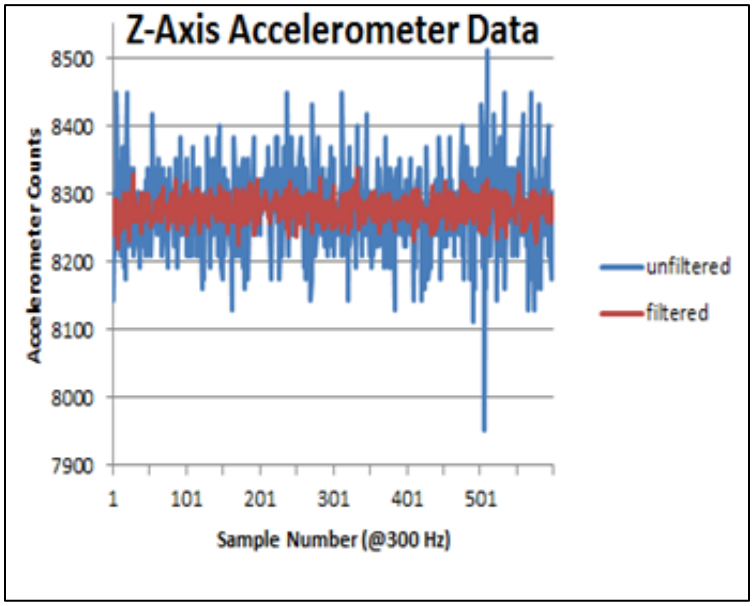


Figure 61: Effect of a running average filter on IMU data

The implementation of this filter caused the polling to run marginally slower, bringing the total read speed from approximately 330Hz to 300Hz. Test data was sampled from the accelerometer as it featured the highest levels of variation: up to 250 counts in either direction. As a result of the filter, the maximum variation of the accelerometer was brought to 50 counts, which is approximately equal to .06 m/s^2 and is sufficiently accurate for the design requirements of IPASS.

IMU communication and PPM signal generation were handled by an Arduino Pro Micro. Because of a conflict in which the Arduino Pro Micro used the same pins for I^2C, the IMU was never integrated into the system as it was deemed less important than motor control. In future designs, a different Arduino would be able to handle both the IMU and motor control.

2.8 Data Transfer

Communications and system integration are a major aspect in designing the IPASS. figure 67 shows the major components of the IPASS and their methods of interacting with each other. For example, the red arrow indicates a USB protocol connection while a black arrow indicates a wireless protocol connection and an orange arrow indicates a PPM protocol connection. The individual blocks are also color coded based on their sub system: green for User Interface, blue for embedded computing, purple for sensing, and orange for propulsion.

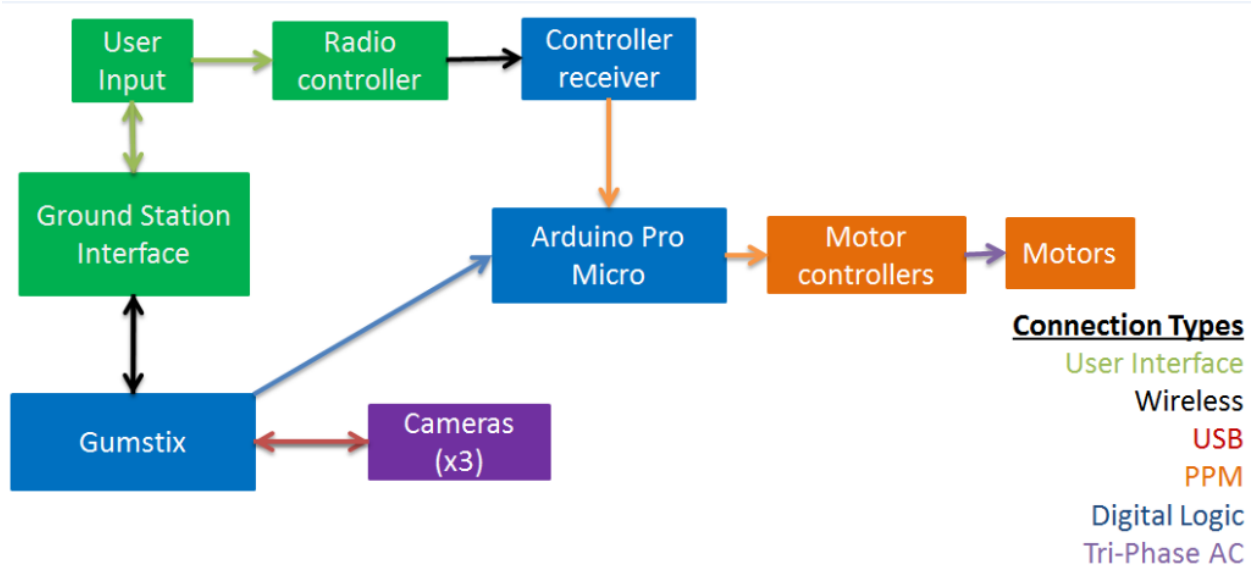


Figure 62: Final block diagram for the IPASS

2.8.1 Ground Station Communication

The Raspberry Pi and the Gumstix are designed to be the main communications hub within the IPASS. The Raspberry Pi has an external wireless card, while the Gumstix has a built in wireless card. The team chose to communicate to the ground station via Transmission Control Protocol (TCP) packets because of its reliability and ubiquity among all internet applications.

Since the IPASS and ground station are connected over a Wi-Fi link, two methods were established to create the connection. Firstly, the IPASS and ground station connects to a wireless network hosted by a wireless router. The IPASS then sends the images to the ground station. The

router's higher output power allows for a greater distance between the IPASS and ground station but would require external power for the router.

The second method of connection is through an ad-hoc network. The IPASS connects to an ad-hoc network hosted on the ground station. This allows the IPASS to send images directly to the ground station without the need for a wireless router.

Both of these methods require a first time setup. However, once the connection is established it does not need to be set up again. The IPASS and ground station will automatically connect to each other. The team decided to use the ad-hoc setup because it does not require a router to be used in conjunction with the IPASS.

A messaging protocol was developed to send command and data messages to and from the IPASS. This protocol utilizes an identifying command token, declared packet size, checksum value, and acknowledgements to ensure that the messages are received and decoded properly. This messaging protocol is demonstrated in detail in Appendix H. All communications outside of image transfers are conducted over this protocol as these messages are designed to be short and light weight. Image transfers are handled by a separate system as they are too large to be handled by the messaging protocol.

2.8.2 Image Transfer

The team decided to transfer the images via SCP since the IPASS was running a full Linux environment. SCP allows for a secure connection between two computers and would ensure a safe and encrypted file transfer.

2.9 Software

This chapter details the software used and developed for the IPASS. The software detailed includes programs used for image processing, ground station GUI, and the embedded computing.

2.9.1 Image Processing

Due to the computation involved in image processing, preexisting image processing libraries are utilized. The background of these libraries and other relevant software is discussed to provide an understanding of the calculations involved in image processing.

Camera Focus

The cameras that were selected have an adjustable focus via a threaded lens. The cameras are mounted with intersecting fields of view so that the image result will have a wider field of view. The selected camera does not have autofocus, so it is required that the lens be focused before the images are captured. The focus was determined manually for the IPASS design requirements of 100 ft. altitude. This was performed by streaming the camera image to the screen of a PC, then aiming the camera at an object 100 feet away and manually rotating the lens in its mount until the image is focused on the object.

Image Stitching

The information stored in the multiple images captured from the multiple cameras on the IPASS will be more difficult for the user to decipher if they are presented alone. A more useful option is to have the images stitched together so as to allow the user one larger image that

removes the common areas in the images and shows all the data from the multiple images in one image.

There are multiple ways to perform image stitching which range in accuracy and computational effort. On one end of the spectrum is simple stitching based on a pre-computed homography for each image. On the other end of the spectrum is homography estimation based on matching features. These features could be common ridges or blobs of pixels in each image [35]. A relatively complicated image stitching pipeline is shown in figure 68 [36].

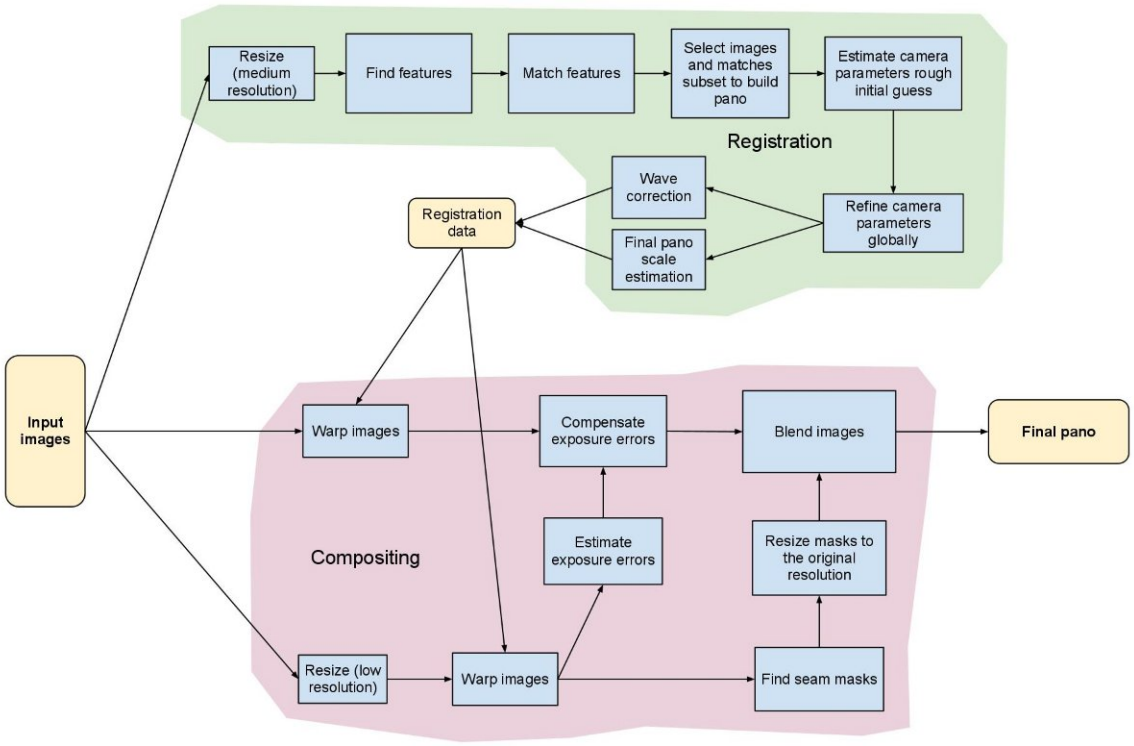


Figure 63: Example automatic image stitching pipeline [36]

One possible way to perform automatic image stitching is through using feature detection and homography estimation. Feature detection can be performed using a scale invariant feature transform (SIFT), which detects features using a difference Gaussian function. The results of this algorithm can be seen in figure 69 [35]. Since there is only one row of images, this is considered 1D stitching. 2D stitching is a more difficult problem, involving multiple rows of images [35].

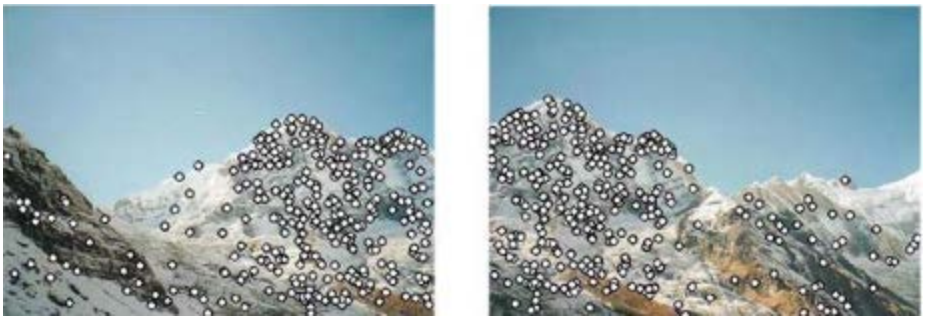


Figure 64: SIFT Matches [35]

After finding the matches, a probabilistic model using a random sample consensus (RANSAC) is used to remove the outlying features in each image that do not have corresponding matches, resulting in features that are shared between images. These matches that are shared between images can be seen in figure 70 [35].

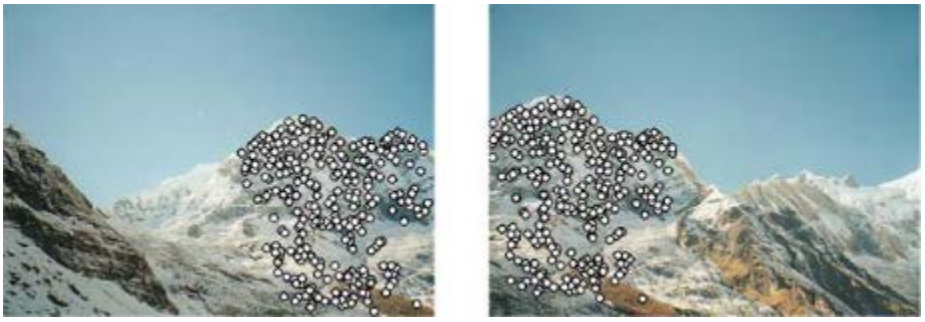


Figure 65: Outliers removed using RANSAC [35]

These features are then used to estimate a homography, or transformation matrix, between images. This transformation can be applied to result in a stitched image. The resulting image can be seen in figure 71 [35].

Figure 66: Resulting stitched image [35]

Another common method for feature detection is called speeded-up robust features (SURF). This method uses a different method for identifying features but will not take into account rotation like SIFT will. SURF is faster than SIFT and just as reliable when using images that are not rotated [37]. figure 72 shows images stitched using SIFT matching and figure 73 shows the same images stitched using SURF [37].

Figure 67: Images stitched using SIFT [37]

Figure 68: Images stitched using SURF [37]

For use in the IPASS, this complicated automatic stitching algorithm is not the best option due to the fact the cameras are always in a fixed position relative to each other and the requirement to perform stitching in real time. It would be inefficient to automatically generate matches and a homography estimation for every set of images captured. This would also result in a lower frame rate when compared to a pre-computed homography based stitching method. The simplified pipelines can be compared in figure 74

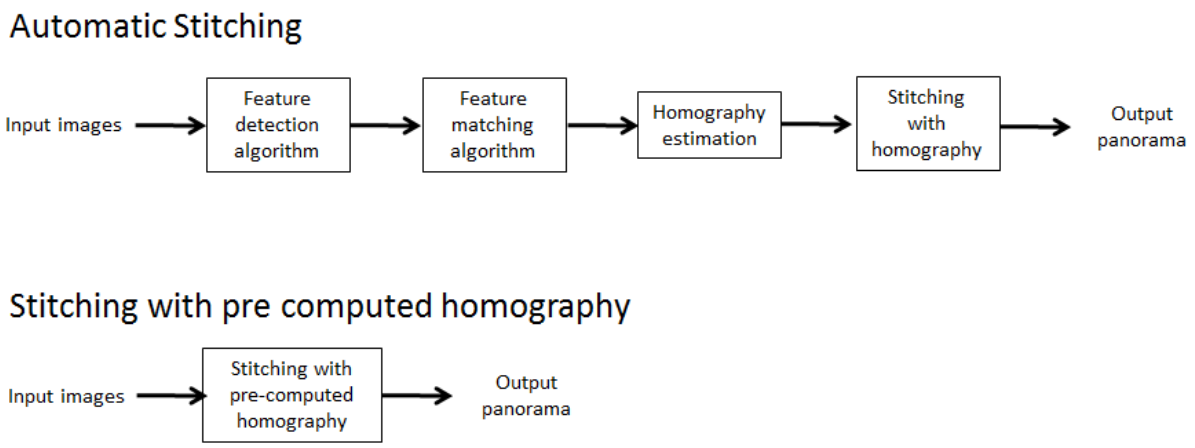


Figure 69: Stitching methods considered

Hugin: Panorama photo stitcher

Hugin was the initial proposed software to be used for image stitching. The ground station would save the images and then use Hugin to stitch them. Hugin is advertised as an easy to use cross-platform panoramic imaging toolchain [38]. The software includes automating features including calibration, automatic stitching, and batch processing. The software is scriptable and allows calling Hugin functionality from Python scripts. There is also the capability of calling Python functionality from within Hugin [39]. These features made Hugin an option considered for the IPASS image stitching subsystem.

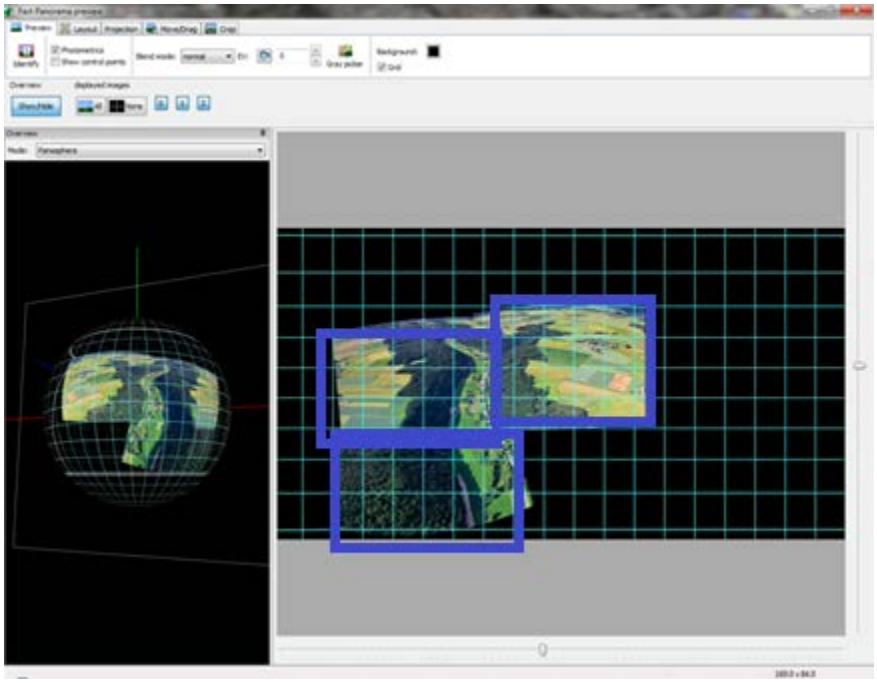


Figure 70: Hugin screenshot stitching 3 images (boxed)

During testing, while using the Hugin software package to stitch multiple images, as seen in figure 75 it was discovered that the automatic stitch routines would lose image data in areas that contained few noticeable details. This can be shown in figure 77, where the 4 images shown in figure 76 were stitched using the automatic stitching feature of Hugin. This example demonstrates 2D stitching, which takes more computational effort than 1D stitching. The layout of the camera on the IPASS at the time required 2D stitching, so this test was useful in determining the feasibility of using Hugin.

Figure 71: 4 images to be stitched

Figure 72: 4 images stitched using Hugin (missing imagery circled)

It can be observed that there is missing image data where there is a scarcity of features such as ridges and objects on the grass field in the image. The automatic stitching routines were also slow. Performing this stitching took approximately 10 seconds on a test laptop. This would have bottlenecked the IPASS image transfer and stitch pipeline if compared to the 1-2Hz desire frame rate. Additionally, it was discovered that the advertised python scripting interface was undocumented and deprecated. Without automation features it was clear that another option would be needed for the image stitching for the IPASS.

JavaCV

A popular software library for image processing is OpenCV. OpenCV includes all necessary functions that the IPASS image processing subsystem would require, including calibration, image transformation, and automatic stitching among others. OpenCV is written in C++, but wrappers exist for C, Python, Matlab, and Java API's as well. Since the ground station software was programmed in Java for cross platform development purposes, it was decided to use the JavaCV wrapper for OpenCV to perform image stitching.

JavaCV includes a library that can automatically generate stitched images much like Hugin. This stitching class uses a matching algorithm to determine where the images overlap. From there the stitching class will create a stitched image from the source images. This process is slower than pre-computed homography based stitching and only works if the source images have enough distinct features to generate a homography. For example, if solid images are cut in software, the algorithm will match the parts perfectly and stitch the cut image back together as long as there are enough matching features. The automatic stitcher in JavaCV requires nearly 40% overlap, along with multiple matching features. This is shown in figure 78.

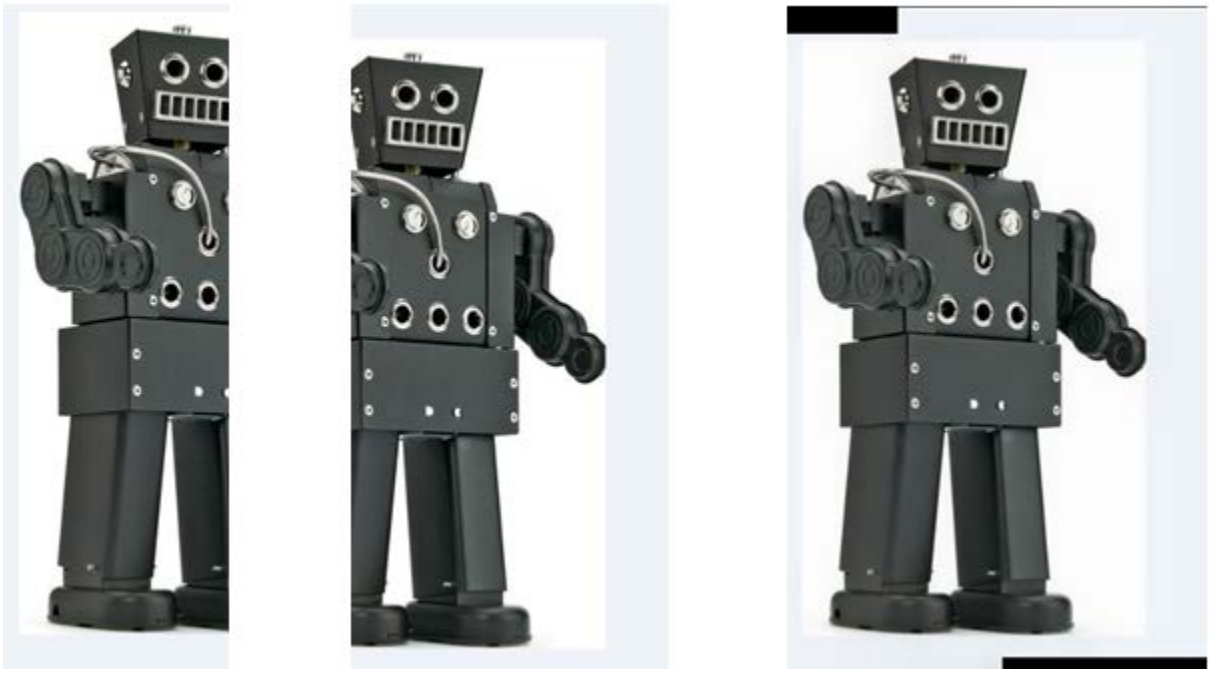


Figure 73: An example of JavaCV auto-stitching. The two images on the left were stitched to make the one on the right.

If there isn't enough detail in the images, the stitcher will fail to stitch the images. The images in figure 79 have the required 40% overlap but the stitcher could not find enough matching features to generate a homography.

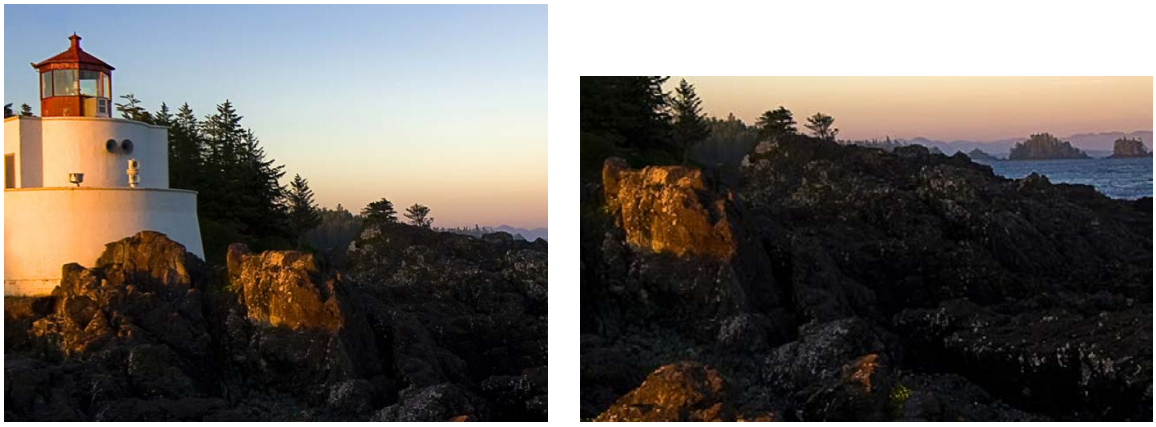


Figure 74: Images with enough overlap but not enough matching features for the default OpenCV stitcher [25]

This automatic stitching will not be suited for the system because its success depends on capturing distinct image data. A more consistent and faster option would be to use stitching based on perspective transforms. A perspective transform uses a transformation matrix to remap the points in an image. This transformation matrix can be generated based on a set of four corresponding points that match between images [41]. OpenCV can generate the perspective transform matrix from four pairs of corresponding points to satisfy Equation 8.

$$\begin{bmatrix} t_i x_i' \\ t_i y_i' \\ t_i \end{bmatrix} = M * \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

Equation 8: Image transformation

Where M is the 3x3 homography matrix that maps the destination points x_i',y_i' and the source points are x_i,y_i for a set of i=0,1,2,3 [41].

This method of image stitching works for rotating images as well. If the source points are selected to be the four corners of the image and the destination points are the new corners with the inverse of the aspect ratio of the source image than the image will rotate 90 degrees. The original image and the result can be seen in figure 80.

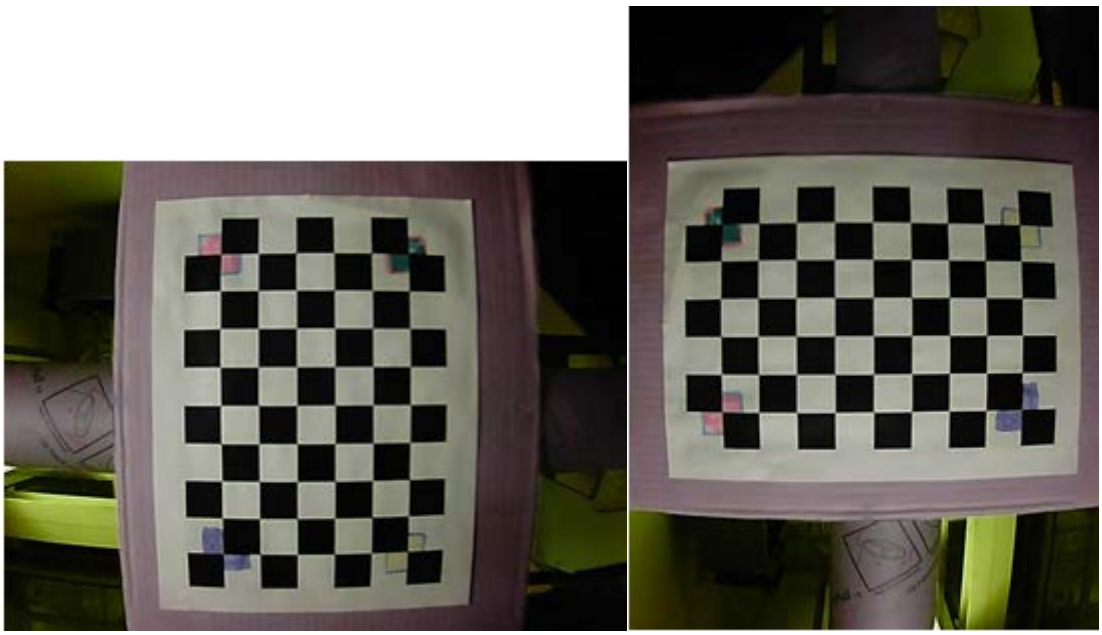


Figure 75: Example of image rotation and calibration in software counterclockwise 90°

A grid with colored corners is used to generate the homographies. The four corners act as anchor points to create the transformation matrices used for image stitching. Choosing the circled points in figure 81, yields the homography:

$$M = \begin{bmatrix} 0.275664 & 0.05499276 & 762.5588 \\ -0.23756132 & 0.9942623 & 23.65323 \\ -7.6214166\text{E}-4 & 5.8408088\text{E}-5 & 1.0 \end{bmatrix}$$

Between the right and center image, and:

$$M = \begin{bmatrix} 1.8367479 & -0.122695476 & 4.7196693 \\ 0.3799376 & 1.2086297 & -67.19927 \\ 0.0011132342 & -2.5864175\text{E}-4 & 1.0 \end{bmatrix}$$

Between the left and center image.

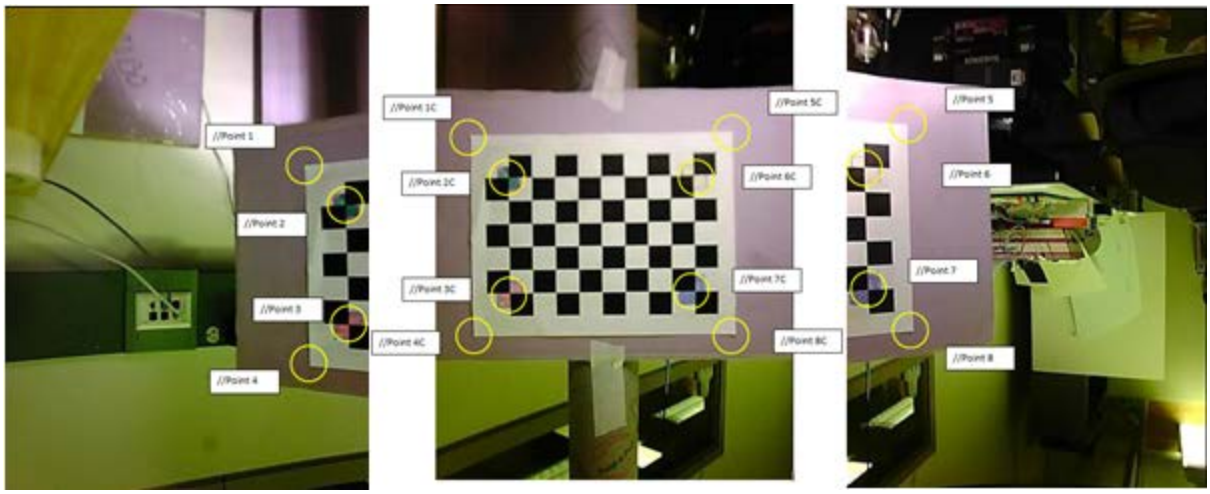


Figure 76: Points for creation of transformation matrices

The overall stitching process used in the IPASS system takes four steps. First, the three images are calibrated, removing the radial distortion caused by the lens. Next, all the calibrated images are rotated 90 degrees to match the orientation of the cameras on the electronics box. Next, a blank image is created that is an appropriate resolution for the stitched image and the center image is translated into the middle of the new blank image. Lastly, the side images are warped onto the stitched image using previously generated transformation matrices. This process can be observed in figure 82.

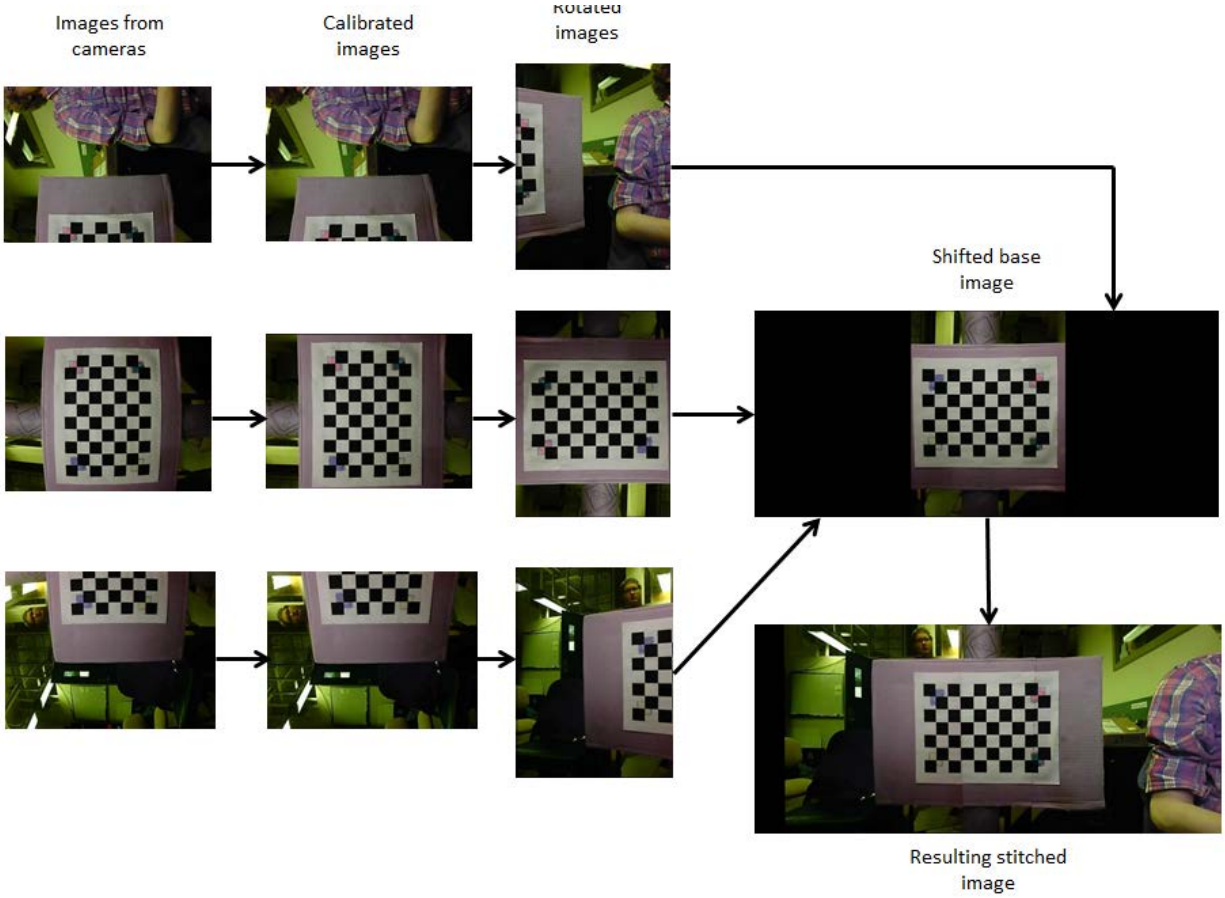


Figure 77: Image stitching process

Camera Calibration

The IPASS cameras are inexpensive CMOS sensors with lenses. These lenses radially distort the image near the edges. Two types of radial distortion can be seen in figure 83. Radial distortion can be addressed in software by applying a transformation matrix to remap the pixels of the image, resulting in an undistorted image. This transformation matrix can be generated and applied using OpenCV [42].

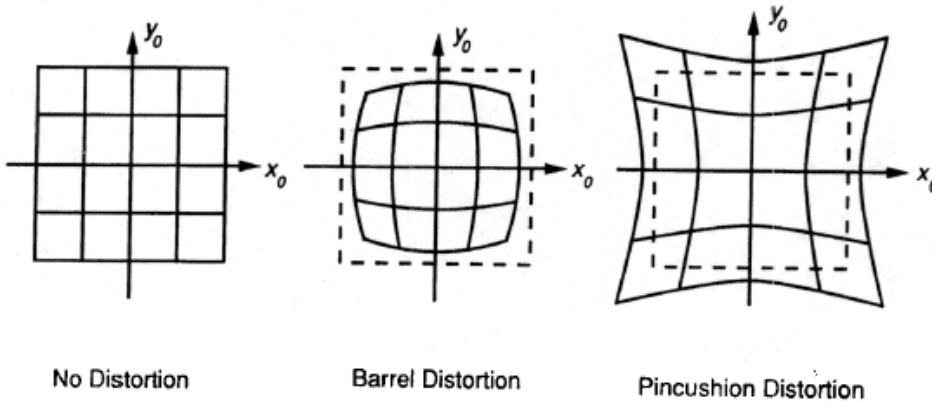


Figure 78: Different types of radial distortion [43]

OpenCV can also take into account the tangential distortion of an image. This phenomenon commonly occurs in low cost cameras when the lens used is not parallel to the plane of the CMOS. OpenCV uses these formulas to adjust for this effect [42]:

$$X_{corrected} = X_{original} * (1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

$$Y_{corrected} = Y_{original} * (1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

OpenCV uses the following formulas for camera calibration [42]:

$$X_{corrected} = X_{original} + [2 * P_1 * X_{original} * Y_{original} + P_2(r^2 + 2 * {X_{original}}^2)]$$

$$Y_{corrected} = Y_{original} + [2 * P_2 * X_{original} * Y_{original} + P_1(r^2 + 2 * {Y_{original}}^2)]$$

In these equations the distortion coefficients are k_1, k_2, p_1, p_2, and k_3, which depend on the lens and focal length. R is the radius of the pixel from the center of the image. OpenCV transforms the image using this matrix multiplication:

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} fx & 0 & cx \\ 0 & fy & cy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Where fx and fy are the focal lengths of the lenses and cx and cy are the center location of the images [42]. To calibrate the camera the transformation was applied to a captured image of a standard grid. The calibration parameters were adjusted manually. The effect of calibration can be seen in the before and after images figure 84 and figure 85, as well as figure 86 and figure 87. Notice that the lines are straighter after the calibration.

Figure 79: Image before calibration

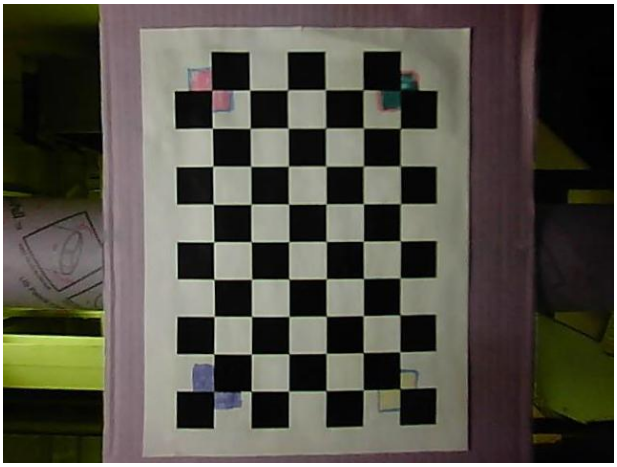


Figure 80: Image after calibration

Figure 81: Image before calibration

Figure 82: Image after calibration

2.9.2 Ground Station Software

The ground station laptop shows a Graphical User Interface (GUI), as seen in figure 88, designed to operate in cooperation with IPASS device. This GUI is developed in Java using JRE1.6 because of the versatility of the Java programming language and its high portability to other systems. The ground station software connects to the IPASS via Wi-Fi and is responsible for sending commands to the device as well as handling image stitching and display.

Figure 83: Startup Screen of the IPASS GUI on an Ubuntu computer

The center of the GUI is populated by a single window. This window displays stitched image data captured from the IPASS device. The device will send three individual images to the ground station using the SCP protocol and then send a packet notifying the ground station of new image arrival. Upon receiving an image notification packet a routine is triggered in software to stitch, save, and display these images. Each image is loaded into memory, stitched together, and then saved into a backup folder. The stitched image is saved into a folder for end-result pictures and then the software displays the most recent stitched image. This process can be seen in figure 89. For more information on image stitching, see 2.9.1 Image Processing.

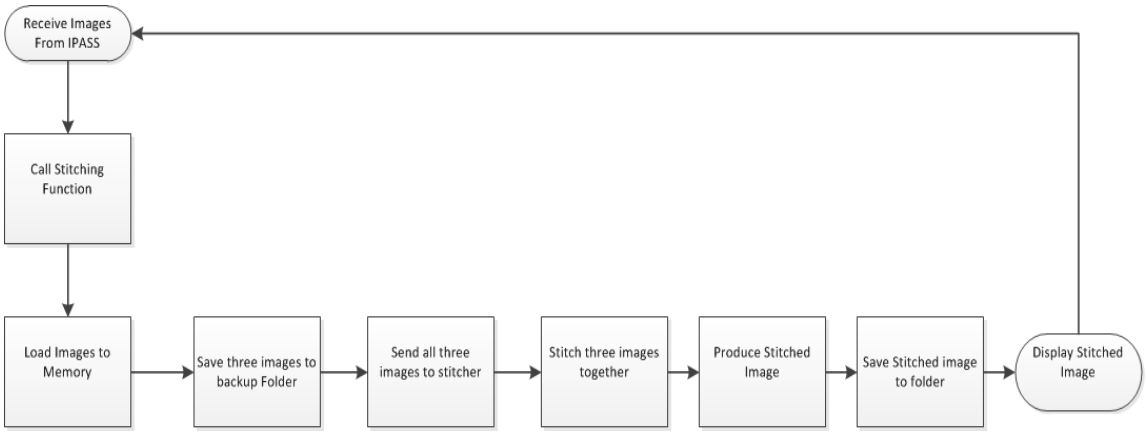


Figure 84: Ground Station Software Flow

The GUI features three buttons on the bottom of the screen; a green Launch button, a yellow Land button, and a Red abort button. When pressed, the Launch button will send the "Launch" packet to the IPASS. When the Land button is pressed, a similar "Land" packet is sent to the IPASS. The Abort button acts as a kill switch for the system, sending the "Abort" packet to the IPASS. Detailed information on the packet protocol is included in Appendix H: Message Protocol..

These three buttons act as the primary control for the IPASS during operation. Upon receiving a "Launch" packet, the IPASS will begin capturing images and enable RC. The "Land" packet will allow the device to continue streaming images, but disables RC. While landing, the motors are forced to half throttle, allowing the device to slow its descent. The "Abort" packet stops all functionality on the device, forcing the motors to brake and stopping image capture. These buttons have been implemented, tested, and proven to operate as described. Every test that captured image data was conducted using this software.

In addition to the three control buttons, there are two buttons labeled "Prev" and "Next". These buttons will change the currently displayed image by the GUI by moving forward and backward in chronological order of images captured by the device. This process is demonstrated in figure 90, figure 91, and figure 92 where team members can be seen approaching the device.

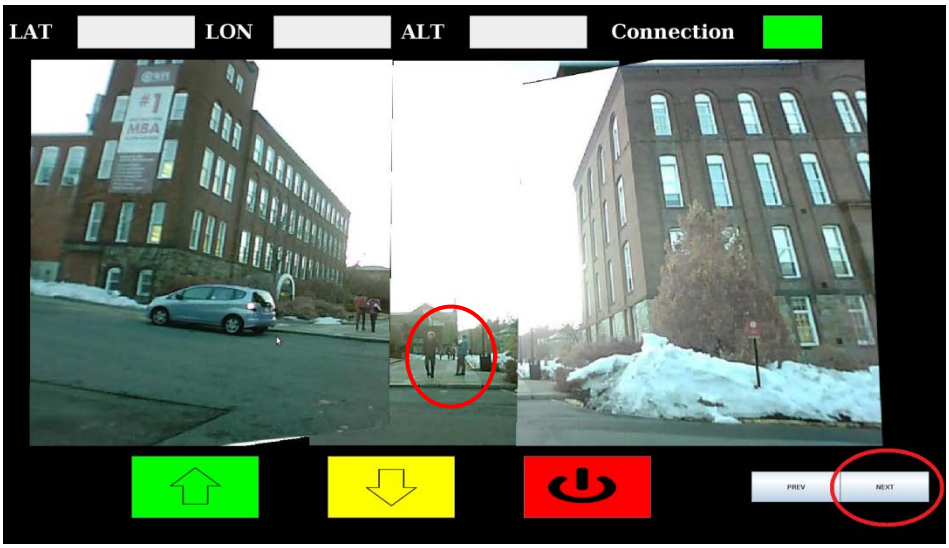


Figure 85: Image browsing functionality. Team members farthest away and circled.

Figure 86: Image browsing functionality. Team members closer.

Figure 87: Image browsing functionality. Team members closest.

In figure 93, figure 94, and figure 95, the images are browsed in reverse order, and the team members are seen moving away from the device.

Figure 88: Image browsing functionality. Team members closest.

Figure 89: Image browsing functionality. Team members farther away.

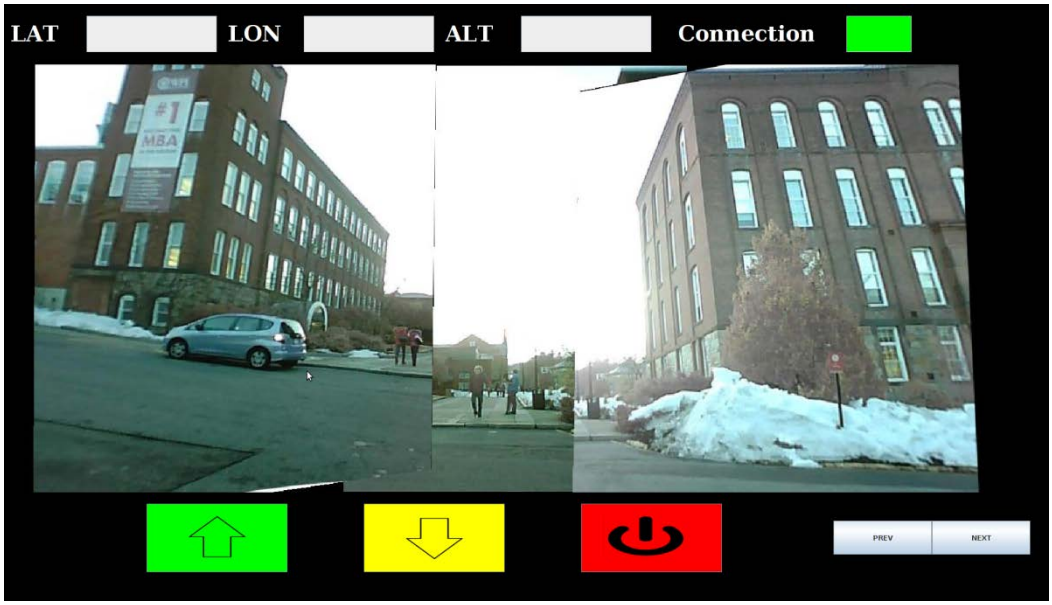


Figure 90: Image browsing functionality. Team members farthest away.

Above the displayed image in the GUI, three text fields are populated with the latitude, longitude, and altitude data gathered from the GPS. These fields are implemented so that they update whenever a GPS packet is received from the device. The GPS data is stored as a .txt file and delivered to the ground station via SCP. The format for a GPS file is shown in figure 96

```
$GPGGA,TIMESTAMP,LATITUDE,N,LONGITUDE,W,1,04,2.7,ALTITUDE,M,-33.8,M,,0000*60
```

Figure 91: GPS data format

This GPS data file is opened in software, and each variable separated by commas is stored in memory. The corresponding fields for latitude, longitude, and altitude are then displayed in the respective fields. This can be seen in figure 97.

Figure 92: GPS data loaded into GUI

To the right of the altitude field is a connection status indicator which will either be red or green. While red, no connection is present. The indicator turns green to notify the user of a successful connection between the IPASS device and the ground station. This effect can be seen in figure 98.

Figure 93: Change in GUI when a connection to the IPASS is established.

2.9.3 Embedded Software

The IPASS control software is designed to operate in cooperation with the ground station. The IPASS is connected to the ground station over a standard Wi-Fi link and is equipped to send and receive the commands outlined in the message transfer protocol. The messages include notifying the ground station that pictures are ready as well as the launch, land and abort commands. The software is designed in ANSI C and is able to be ported to other computers for testing or for other IPASS iterations. This software is responsible for sending and receiving commands to and from the ground station, gathering image data from the on board cameras, and sending thrust control information to the Arduino. The software also contains generated Doxygen documentation to expedite familiarity.

Once in operation, the software follows the flow chart in figure 99. First the software waits until it is able to establish a connection from the ground station. Once the connection has been created the IPASS will start transmitting GPS data to the ground station. The IPASS software will not gather image data until the operator has instructed the IPASS to launch. When the software receives the Launch command it enables the picture taking and transmitting subroutines while also enabling the Arduino which begins powering the propulsion system. The software remains in this state until the operator tells the IPASS to land or abort. When a land procedure is initiated the software continues to gather and transmit image data but instructs the Arduino to slow the propulsion so that the IPASS will land. If an abort procedure is initiated the software will cease gathering image data and the Arduino will stop controlling the propulsion system.

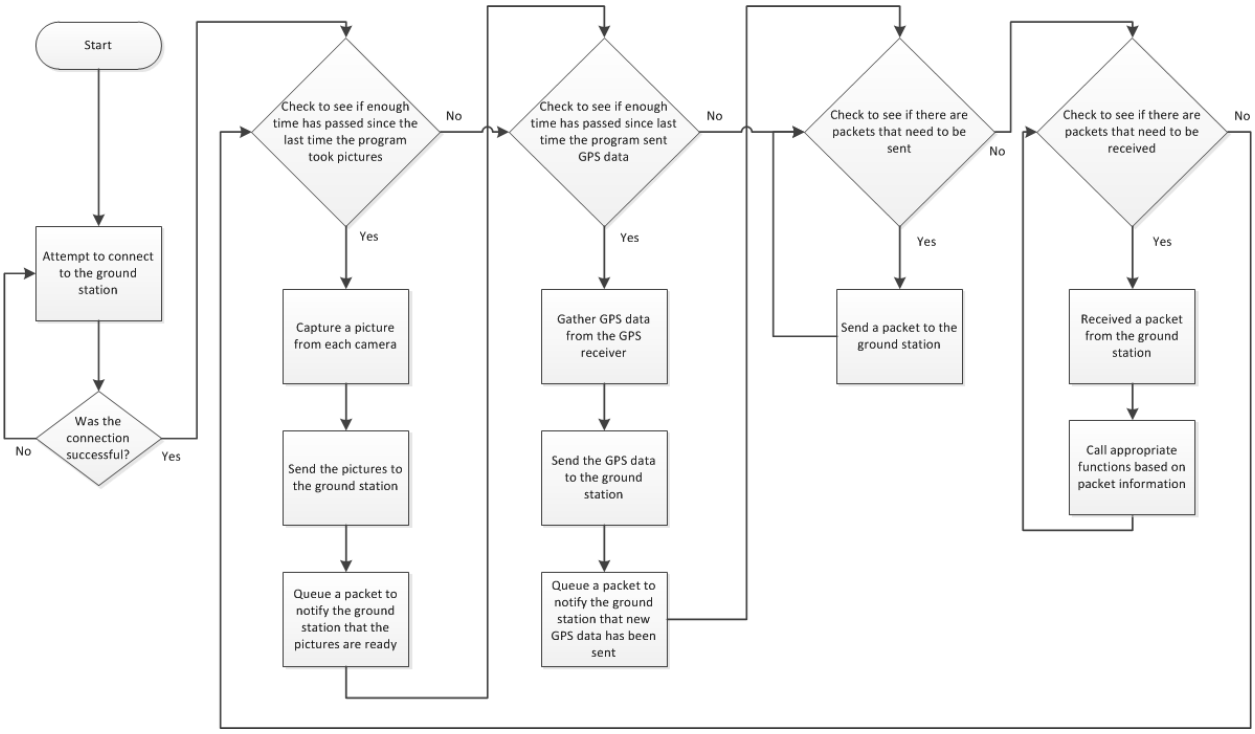


Figure 94: Flow chart for the embedded computing software

During image gathering the IPASS takes three pictures from its onboard cameras, then sends the images to the ground station via the SCP command and finally it notifies the ground station that it has sent the images. During GPS transmission the IPASS gathers GPS information from the GPS receiver sends the information via SCP to the ground station and then notifies the ground station that the GPS information has been sent.

2.10 Ground station

A key part of the IPASS's operation is the process of stitching camera data and displaying the resultant image to the user. This process is accomplished by a ground station that receives the camera data wirelessly transmitted from the IPASS. In order to ensure an ease of use and quick update of surveillance data a dedicated ground station is required. Specific requirements included an anti-glare screen, at least 2.0 GHz of processing power, and a solid state drive.

The Toughbook brand of laptops fits many, if not all, of these requirements. As Toughbooks are already used in a variety of military applications, the team researched their potential for use with the IPASS. The fully ruggedized and semi-ruggedized Toughbook models fit the requirements for the ground station but were prohibitively expensive. With prices ranging from two to three thousand dollars, the ground station could cost more than triple the manufacturing cost of the IPASS.

The Lenovo Thinkpad W530 was chosen as a ground station for the IPASS because it has comparable capabilities to the Toughbook models at approximately half the cost. The Thinkpad costs approximately sixteen hundred dollars, which, while still double the cost of the IPASS, was well within the project's budget. The Thinkpad also has the potential to be used in future

projects. Specifications for the Thinkpad can be seen in Appendix L: Lenovo Thinkpad Specifications.

3. Results

In this section the team's accomplishments are evaluated with regards to the IPASS design requirements. The degree to which these design requirements are accomplished is analyzed to provide an analysis of the effectiveness of the IPASS.

3.1 Summary of Accomplishments

The final IPASS design is capable of receiving commands from the ground station and sending the ground station data packets. After receiving a launch command, the IPASS is controlled remotely for launch. During flight, the IPASS flew about 15 ft. high before crashing. Upon crashing, the internal systems maintained functionality and continued to send data to the ground station.

The IPASS is able to receive commands telling it when to start taking pictures from the three cameras. Once this command is received, the IPASS continually sends pictures to the ground station for stitching and display. A new stitched image appears on the ground station once every 3-5 seconds. The electronics box is capable of sending and receiving data independently from the IPASS chassis and has the potential to be used in other applications.

As described in 1.4 Design Specifications, the IPASS needs to meet several requirements. Table 8 details which IPASS design requirements were met, not met, or conditionally met.

Table 8: Table of results

Design Requirement	Degree of Success
Reach height of 100 ft.	Not Met
Survive a fall of 30 ft.	Met
Weigh less than 20 lbs.	Met
Method to retard fall	Not Met
Capture of useful image data	Conditionally Met
Location sensing	Conditionally Met
Functional embedded computing	Met
Transmit visual and location data	Conditionally Met
Transmit wirelessly with 200 ft. range	Conditionally Met
User friendly	Met

Reach a height of 100 ft.

The IPASS never achieved a height of 100 ft. during a launch test. The IPASS flew approximately 15 ft. maximally. The IPASS was unable to achieve this height due imbalances in the system and vulnerability to wind.

Survive a fall of 30 ft.

A fully operational IPASS was dropped from 30ft. two consecutive times and the ground station still continued to receive images from the cameras. The IPASS chassis itself sustained some minor structural damage. The internal components of the electronics box remained undamaged but the Gumstix became unseated, causing a loss in connection to the ground station. Full details of this free-fall test can be found in Appendix K: IPASS Survivability Test.

Weigh less than 20 lbs.

The final IPASS deice weight is 4.4lbs. and is well under the maximum weight requirement.

Method to retard fall

The team has implemented methods for the IPASS to slow itself during descent. This requirement has not been met because the IPASS slow-fall routine remains untested.

Capture useful image data

During electronic box testing, the IPASS was able to capture, send, stitch, and display images with a resolution of 1440 pixels x 640 pixels in 3-5 seconds. In these pictures, humans are clearly visible at distances up to 100ft. Images from various electronics box tests can be found in Appendix F: Electronics Box test. Because the full system test only resulted in a flight height of about 5ft, the image data gathered proved not to be useful.

Location sensing

The requirement to sense location data was to be fulfilled by a GPS receiver and IMU. The GPS receiver was successful in transmitting location data to the ground station from the IPASS. However, due to power problems, the cameras and GPS could not both be used on the IPASS at the same time. The Gumstix could not produce enough amperage to provide sufficient power for three cameras and for the GPS receiver. The team decided that the cameras were a more important aspect of the project and the GPS was not implemented during full system testing. In a similar fashion a pin conflict was discovered during testing that prevented the Arduino Pro Micro from generating motor control signals and communicating with the IMU simultaneously. Maintaining motor control was more important to the operation of the IPASS than the integration of an IMU so the IMU was not implemented.

Functional embedded computing

The Gumstix and Arduino Pro micro are successful in gathering image and location data, communicating information to and from the ground station, and controlling when the user could operate the motors.

Transmit visual and location data

As stated previously in this section, the IPASS can capture both useful image data and location data. While both sets of data cannot be transmitted simultaneously or sequentially due to power restrictions, they can both be transmitted individually.

Transmit wirelessly with a 200 ft. range

The 200ft. wireless range was chosen as a somewhat arbitrary distance for transmitting data. During electronics box testing, the team discovered that the range of the ad-hoc network between the Gumstix and the ground station has a range of approximately 120ft. with a noticeable drop in bandwidth at about 75ft. The team deemed this range sufficient for operation and did not further pursue a 200ft. wireless range. Details of this test can be found in Appendix I.

User friendly

The IPASS is designed to be easy to use so that operations could be carried out in high-stress environments. First-time setup of the IPASS is non-trivial but need not be done in the field. This set up process along with instructions for operation of the IPASS can be found in Appendix O: IPASS Instruction Manual. Construction of the IPASS chassis can be done quickly with no tools. In-field operation requires only three terminal commands: launch, land, and abort. Operation of the user interface is intuitive due to easy to understand buttons and simple operation.

The team also purchased a Pelican 1560LFC Overnight Laptop case for the storage and transport of the IPASS. The Pelican 1560LFC was chosen because it is watertight, crushproof and its 1.64 cu ft. of storage space has enough room for the entire system including the ground station laptop. The Pelican 1560LFC ensures that no accidental damage will occur to the IPASS during transport and storage. The IPASS can be seen deconstructed and stored in the case in figure 100.

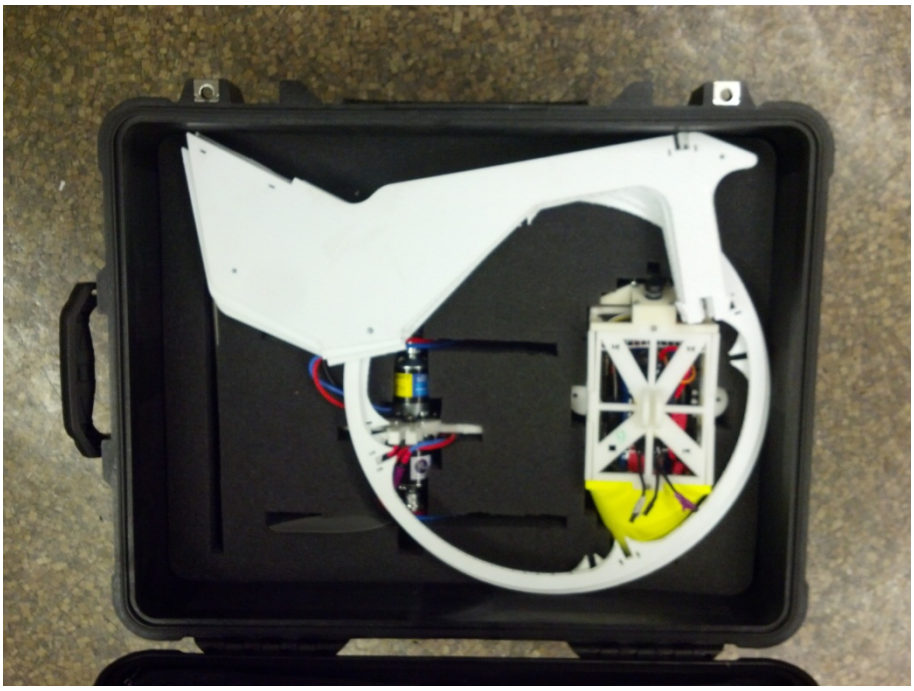


Figure 95: Pelican case with stored IPASS

3.2 Discussion

The maximum height reached by the IPASS during flight was about 15ft. The IPASS was unable to fly any higher due to being off-balance and vulnerable to wind. To help mitigate this issue, the team experimented with contra and co-rotating propeller setups. Contra-rotating propellers provide more lift to the IPASS but make it too off-balance to leave the ground; the motors would be set to maximum thrust and the IPASS would just fall over.

Co-rotating propellers provide less overall lift, but make the system more balanced. In this configuration, the propellers would induce spin on the IPASS causing it to rotate as it launched, centering its mass and making the overall system more stable. The team discovered

that because the system was so light, the additional lift from contra-rotating motors was not needed and instead opted for more stable flight rather than additional thrust.

Taking a picture from a single SB101C camera module using MPlayer took approximately 1.5 sec. For the Gumstix to take, save, and send three pictures it took 3-5 seconds. The ground station saved and stitched the three images in less than a half a second. Sending each image took less than a tenth of a second at ranges from 0 to 70 feet.

Saving the images to the Gumstix proved to be the bottleneck in the system. To mitigate this, the team attempted to take multiple images with the Gumstix concurrently. The team discovered that the Gumstix's video drivers are not capable of doing this and will respond by only capturing one image correctly and then sending the correct image and nothing for the other two.

Since the image stitching was done with homographies, any camera movement during operation would cause the stitched images to appear slightly off. This hurt the overall system reusability. After a landing, the cameras would move and if the system was tested again, the resulting images would appear offset. An example of this offset can be seen in figure 101.

Figure 96: Stitching offset caused by camera lens movement.

4. Conclusion and Recommendations

IPASS has established itself as a proof of concept as a surveillance system. The IPASS never achieved a self-propelled flight high enough to provide useful image data. Chassis imbalances prevented the IPASS from stable flight, though liftoff was achieved on multiple occasions. IPASS control systems have been established but further research into lightweight propulsion mechanisms and stability control is required for optimal performance.

Despite shortcomings in propulsion, the team has made significant strides towards a fully integrated standalone visual sensor package. The electronics box is self-powered, can retrieve and send visual data, and is modular featuring optional GPS and IMU integration. With modifications to the power system and the embedded computing additional sensors may be integrated for use in other applications.

IPASS consists of two primary components: the aerial chassis and the ground station. The ground station laptop features an intuitive GUI capable of receiving, storing, and displaying data received from the system. IPASS has integrated a propulsion system, electronics box, and ground station providing a foundation for further advancement in man-portable aerial surveillance systems.

4.1 Future Work

A primary goal of future development should be focused on the propulsion system. The current system was unable to reach the goal of 100ft due to imbalance issues and vulnerability to wind. Future systems should improve stability to allow sustained flight and be robust enough to maintain flight with gusts of wind.

To improve IPASS stability, controlled fins on the IPASS could redirect thrust from the propellers to stabilize and steer the system during operation. This could be implemented in conjunction with an IMU so that the IPASS would be able to detect its pose in the air and then autonomously actuate its fins to self-stabilize. In order to make the IPASS less vulnerable to wind, the chassis could be made of a heavier but equally protective material. This would make the IPASS physically more difficult to be blown around during operation. However, this may also require stronger motors to provide adequate lift to the system.

Future systems should improve frame rate of the system; the current system displays one image every three to five seconds, far below the design goal of one fps. At the current frame rate, motion of the device causes visual distortion in images that may cause confusion during operation. Future work should remove the bottleneck of saving images to the Gumstix. The suggested approach would be to use ANSI C to gather image data from the cameras directly. Newer implementations should not save images locally to the Gumstix. The process could be further improved by threading the image taking process and creating three child threads to take pictures from each camera concurrently while the main process would be responsible for maintaining communications.

A faster vision system would allow for the IPASS to utilize co-rotating propellers without affecting image quality. In a system where the IPASS rotated during flight, the slow process of taking pictures causes the pictures to appear disjointed when stitched. Being able to take all three pictures at once would allow for the system to move and rotate in the air without degrading picture quality.

As discussed previously, changes could be made to the power system and the embedded computing to allow for full integration for the GPS receiver and IMU in conjunction with the three cameras and motor control. The integration of sensors, stable flight, and higher frame rate will fulfill the initial design goals and bring completion to the project.

5. Project Expenses

For the design and realization of IPASS, the team was provided a proposed budget of $8,000 by the AFRL through OAI. The team reserved $2,000 of this total budget for travel to WSU to present the IPASS, allowing for $6,000 to be spent on development.

The team's expenditures for each major subsystem are detailed in Table 9: Expenditure by subsystem. See Appendix N: Full Expense Report for a full expense list and Appendix M: IPASS Cost Breakdown for a detailed cost of the IPASS used in testing. Multiple parts were purchased in the event of component failure.

Table 9: Expenditure by subsystem

System	Cost
Propulsion System	$751.39
Mechanical System	$1,125.20
Embedded Computing	$1,231.52
Sensing	$707.66
Ground Station	$1,878.00
Pelican Case	$ 274.68
Total Spent	**$5,968.45**
Total Remaining	**$2,031.55**

Propulsion costs included motors, propellers, batteries, wires, and connectors for these components. The mechanical costs included Delrin, Coroplast, acrylic, carbon fiber rods, assembly materials, and 3D printing of the top cones and camera mounts. Embedded computing costs covered the Raspberry Pi and its wireless adapter, Arduinos, Gumstix and its expansion boards, and the SD cards. Sensor costs included cameras, IMUs, and the GPS receivers. The ground station consisted of one Lenovo Thinkpad laptop. The project was under budget by $306.23 after estimated travel costs.

6. Acknowledgements

The IPASS project team would like to give thanks to the following people and organizations. We would like to thank Richard van Hook, AFRL, and OAI for providing us with the opportunity to work on this project. We would like to thank Professor Padir and Professor Lai for advising our project and for their endless support. Thanks to Professor Looft for granting us the space to work in his lab and to RJ Linton and Velin Dimitrov for help with the cameras, Gumstix, and everything else Linux. We would like to thank Ruixiang Du for his advice and his Matlab expertise. We would also like to thank Tracey Coetzee for fielding our numerous part orders and the WPI Robotics Department for preparing us to work on this project.

This material is based on research sponsored by Air Force Research Laboratory under agreement FA8650-09-2-7929. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes not withstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsement, either expressed or implied, of Air Force Research Laboratory, the U.S. Government, or OAI.

7. Authorship

1. Introduction: Written by Adam Blumenau, Alec Ishak, Brett Limone, Zachary Mintz, Corey Russell, Adrian Sudol
2. System Development: Written by Adam Blumenau, Alec Ishak, Brett Limone, Zachary Mintz, Corey Russell, Adrian Sudol
3. Project Expenses: Written by Adam Blumenau, Alec Ishak, Corey Russell
4. Results: Witten by Adam Blumenau, Alec Ishak, Corey Russel
5. Conclusions and Recommendations: Written by Alec Ishak, Corey Russel, and Adrian Sudol

Appendix A: Written by Zachary Mintz
Appendix B: Written by Brett Limone, Corey Russell
Appendix C: Written by Alec Ishak, Zachary Mintz, Adrian Sudol
Appendix D: Written by Brett Limone, Zachary Mintz
Appendix E Written by Adam Blumenau, Alec Ishak
Appendix F: Written by Adam Blumenau, Alec Ishak
Appendix G: Written by Adam blumenau, Corey Russel, Adrian Sudol
Appendix H: Written by Brett Limone
Appendix I:Written by Alec Ishak, Corey Russell
Appendix J Written by Adam Blumenau, Alec Ishak
Appendix K: Written by Alec Ishak
Appendix L: Written by Adam Blumenau, Alec Ishak
Appendix M: Written by Alec Ishak
Appendix N Written by Adam Blumenau, Alec Ishak, Brett Limone, Zachary Mintz, Corey Russell, Adrian Sudol
Appendix O: Written by Alec Ishak, Corey Russell

8. Bibliography

[1] J. G. Chizek, "Military transformation: Intelligence, surveillance and reconnaissance," Congressional Research Service, *The Library of Congress*, 2003.

[2] J. Pappalardo. “The future for UAVs in the U.S. air force,” *Popular Mechanics*. 2010.

[3] S. Tadokoro, “Rescue Robotics: DDT Project on Robots and Systems for Urban Search and Rescue,” London: *Springer-Verlag*, 2009.

[4] D. Wyllie. “Police UAVs: Nearly limitless potential,” 2012

[5] “UAV platform categories,” *UAV business review*. 2012

[6] “MQ-9 reaper,” *U.S. Air Force*, 2012

[7] “$143M for global hawk cost overruns,” *Defense Industry Daily*, 2012

[8] “Analysis of the fiscal year 2012 Pentagon Spending Request,” *Cost of War* 2012

[9] “RQ-11 Raven Unmanned Aerial Vehicle, United States of America,” *army-technology.com* 2012

[10] “Honeywell: T-hawk Micro Air Vehicle (MAV), United States of America,” *army-technology.com* 2012

[11] R. Wheeler, J. Pate, M. Junkerman, C. Beyer, A. Smith, A. Szabo, V. Bhattacherjee and B. Walker, "PALODE – PORTABLE AERIAL LAYERED-SENSING ORDINANCE Final Project Report,"

[12] “Estes Model Rocket A8-5 Engine,” *Hobbylinc* 2013

[13] “The Best Brushless Motors,” *Sky High Hobby* 2013

[14] J. Dickey, “Static Thrust Calculation,” *Quad Copter Project* 2013

[15] E. Strickland, “How a Maple Seed Twirls and Whirls and Stays Aloft,” *Discover* 2013

[16] “Physical properties of Acrylite FF Acrylic Sheet,” AcryliteFFDataSheet.pdf., *Cyro Industries*, 2012

[17] “Material Safety Data Shee,” *Coroplast*, http://www.coroplast.com/technicalinfo/msds.htm. 2012

[18] “The Drag Coefficient,” *National Aeronautics and Space Administration*, 2013

[19] "Gumstix Alcatraz," https://www.gumstix.com/store/product_info.php?products_id=281, *Gumstix*, 2013

[20] "Gumstix Tobi," https://www.gumstix.com/store/product_info.php?products_id=230, *Gumstix*, 2013

[21] "C329-SPI-board JPEG Compression VGA Camera Module (no lens)," http://www.electronics123.com/s.nl/it.A/id.3011/.f, *Electronics123.com Inc.*, 2013

[22] "Gumstix Caspa FS," https://www.gumstix.com/store/product_info.php?products_id=254, *Gumstix*, 2013

[23] "SB101C USB CMOS Board Camera Module," http://www.electronics123.com/s.nl/it.A/id.3263/.f, *Electronics123.com Inc.*, 2013

[24] "Geometric image Transformations," *opencv v2.1 documentation*, 2013

[25] "Camera calibration With OpenCV," *OpenCV 2.4.4.0 documentation* 2013

[26] "Radial Distortion Correction," http://www.uni-koeln.de/~al001/radcor_files/hs100.htm, 2013

[27] J. Meyers, "Japan's Amazing Flying Sphere Can Be Used At Disaster Sites And In Anti-Terrorism Operations", *Business Insider*, 2011

[28] J. Mayness, "Voyeur: Challenging the Imagination", *Unmanned System*, 2007

[29] R. Metx, "Bouncing Gets into Dangerous Places So People Don't Have To", *MIT Technology Review*, 2012

[30] B. Coxworth, "senseFly set to release eBee industrial UAV", *Gizmag*, 2013

[31] "AFRL Student Challenge", http://www.afrlstudentchallenge.org/

[32] "Unmanned Aircraft Systems (UAS): Regulations & Policies", *Federal Aviation Administration*, 2012

[33] "UK deploys toy-sized spy drones in Afghanistan," *RT*, 2013

[34] B. Coxworth, "Lockheed Martin's Samurai monocopter – you won't believe how this thing flies," *Gizmag*, 2011

[35] Luo Juan and Oubong Gwun. "SURF applied in panorama image stitching," presented at Image Processing Theory Tools and Applications (IPTA), 2010 2nd International Conference 2010

[36] “Hugin - Panorama Photo Stitcher,” *Sourceforge.net*, 2013

[37] “Hugin Scripting Interface,” *PanoTools: Next Generation* 2012

[38] “JavaCV stitcher.stitch method throws assertion failure,” *javacv java interface to OpenCV and more* 2013

[39] “Geometric Image Transformations,” *opencv v2.1 documentation* 2013

9. Appendix

Contents

Appendix A: Safety Manual of IPASS

In the event of an Emergency call: **911**

This manual has been developed to ensure the safe testing and handling of team IPASS' Unmanned Aerial Vehicle (UAV) system. UAVs are inherently dangerous and should be handled with care. The UAV used by team IPASS is powered by electric motors that have enough power to cause serious injuries to those who are not careful around the drone. However, if the proper safety and handling of the UAV is partaken by the members of team IPASS and all observing parties, safety can be maximized. The details for safe handling of the drone are outlined in this document.

Every member of team IPASS must read and understand the guidelines outlined in this document before any tests or demos are performed with the drone.

UAV State definitions

These are the different states that the robot will go through during testing.

Name	System State	Personnel Allowed near Drone	Notes
Cold	None: Battery Disconnected	Anyone	Used for maintenance
Warm	Battery Connected, Initial system power up	Only members of team IPASS	No signal to the motors but still keep hands clear
Hot	Fully powered system and transmitter turned on	Only the flight Controller	No hands or personnel near the drone

Safety Procedures for drone testing

1. Before the test the flight controller, who is in charge for the duration of the test, needs to go over all the details of the test will all participating parties, including observers, so that everyone know what is to be tested and where they should be during the test.
2. After everyone has been briefed the drone should be assembled as needed and tethered, if necessary, to prevent the drone from going too far out of control. During this setup the drone is to remain in a **COLD** state because personal will be close to the drone.
3. Once the drone is fully constructed the flight controller should clear the area near the drone of all unnecessary personnel. The persons that are not needed near the drone should move to a safe distance away from the drone as to prevent injury.
4. Once the drone is clear of unnecessary personnel the battery should be connected bringing the drone to a **WARM** state.
5. After the drone is in the **WARM** state no personnel should enter the immediate area above or around the vicinity of the drone. At this point the flight controller should focus only on the drone as to prevent any potential injures.
6. When the flight controller deems the area to be safe and clear he or she may then turn on the drones RC transmitter and bring the drone to the **HOT** state.
7. With the drone in the **HOT** state the necessary tests that are needed to be done may be performed by the flight controller. If at any time the flight controller feels that the drone

is not performing as expected or something fails he or she should cut the power to the drone's transmitter thereby cutting the control signal to the drone and disabling it.

8. After the tests are completed the flight controller should turn off the transmitter and bring the drone back to the **WARM** state. Once the drone is fully in the **WARM** state the flight controller should disconnect the battery bring the done down to the **COLD** state.
9. Once the flight controller has fully disabled the drone all personnel are allowed back near the robot for clear up and data analysis.

Appendix B: Matlab Rocket Feasibility Script

```
%%inputs
mass=3;%in kg
avgthrust=34;%% avg thrust of motor in N, form datasheet
burntime=1.7;%% in s

yburn=.5*(avgthrust/mass)*(burntime)^2;
vburn=(avgthrust/mass)*burntime;

%output in meters
maxheight=-.5*9.8*(vburn/9.8)^2+vburn*(vburn/9.8)+yburn;
```

Appendix C: Camera Resolution Test

Objective

The purpose of this test was to determine at what distance a human can be identified using different camera resolutions. IPASS requirements state that the system needs to be able to send useful data to the user. It was determined that useful data would include being able to identify a human at about 100 feet. The goal was to find the smallest resolution at which a human could be identified at 100 feet.

Methods

To conduct this test, one team member stood 50 yards away while another team member stood at various distances away. At each distance, a picture was taken at different resolutions. A complete list tested resolutions can be found in

Table 10. The distances that were used are in Table 11.

Table 10: Tested resolutions

Resolution
640x480
800x480
1024x768
1600x960
1600x1200
2048x1536

Table 11: Tested distances

Distances
100 ft
150 ft
200 ft
300 ft

Once all the photographs were taken, they were sorted by distance. At each distance, it was determined whether or not the human in each photograph could be identified.

Images Captured

Only images of the subject at 100 and 150 ft. are reported. All other images were used as reference images and did not aid in selecting a resolution. Images above a 1024x768 resolution are not shown as they are too large to fit on the page and would need to be scaled down and therefore would provide no new information.

Figure 97: Human pictured at100ft in a 640x480 pixel image

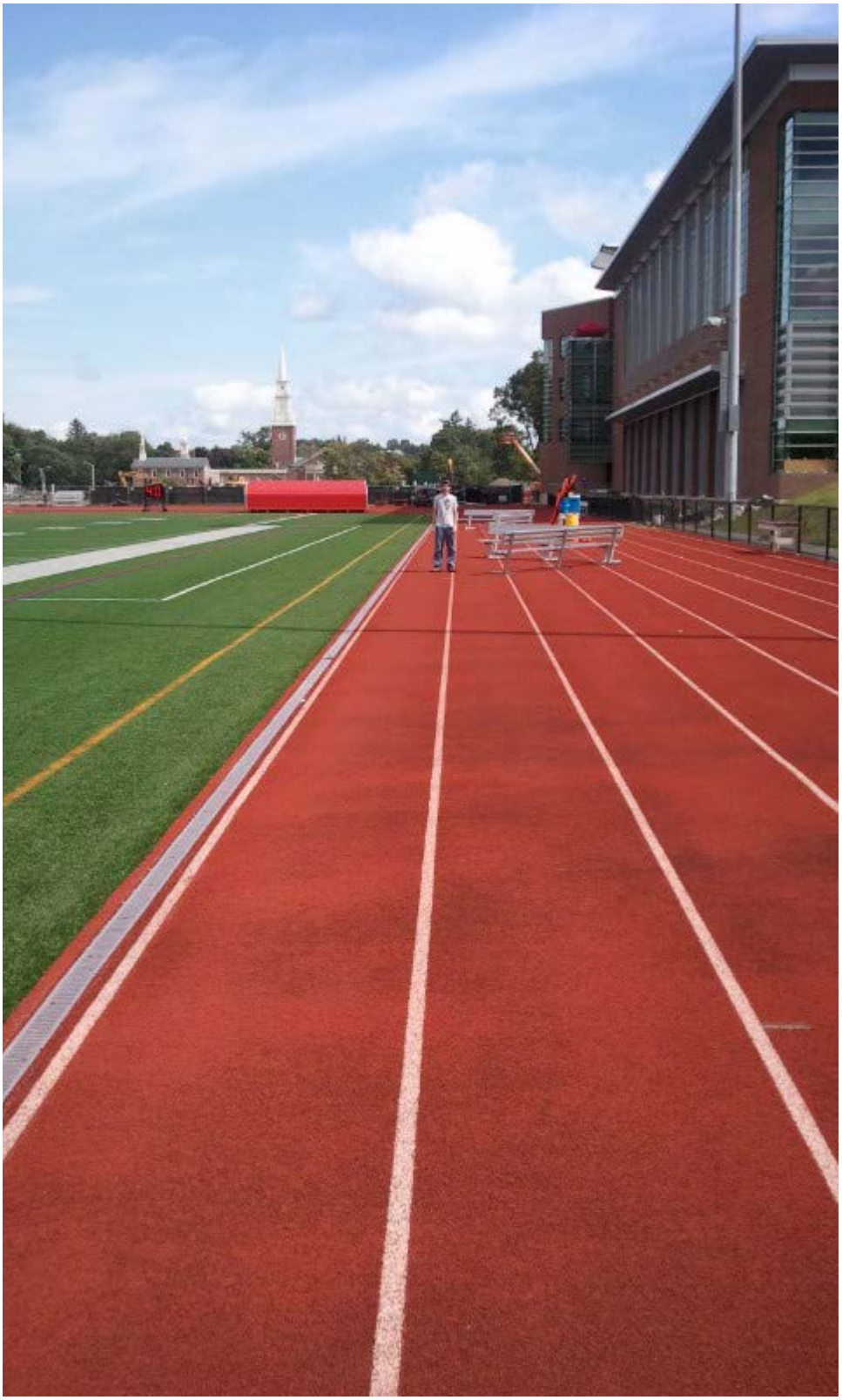


Figure 98: Human pictured at100ft in an 800x480 pixel image

Figure 99: Human pictured at100ft in a 1024x768 pixel image.
This image has been scaled down to fit on the page.

Figure 100: Human pictured at150ft in a 640x480 pixel image.

Figure 101: Human pictured at150ft in an 8000x480 pixel image.

Figure 102: Human pictured at150ft in a 1024x768 pixel image.
This image has been scaled down to fit on the page.

Results

The team visually inspected each series of pictures taken at each distance. To keep the total amount of data that would need to be handled by the IPASS low, the smallest resolution where the human was visible would be acceptable for the IPASS. In the 640 by 480 pixel image, the human in the picture was easily distinguishable at 100 ft. Therefore, cameras capable of taking 640 by 480 resolution images would be used on the IPASS.

Appendix D: IPASS Drop Tests

Introduction

This test was performed to determine whether the material called Coroplast (corrugated polyethylene) would be feasible for use in the construction of the chassis of the IPASS.

Figure 103: Chassis used in the drop test

Assumptions

For this test it is assumed the final IPASS would withstand multiple drops. One requirement of the project is for the robot to survive a 30 ft. drop without having any of the critical components damaged or broken. Most electronic components can easily withstand high accelerations and, provided they do not impact the ground directly, should survive. Component mass was simulated for this test.

Goals

The level of damage sustained by chassis will determine the success of this test. If damage comprises structural integrity, the material will not be used. If there are no breakages or minimal damage then the result is a success.

Measurements

The team was measuring the condition of the materials before and after dropping. Any cracks, breaks or deformations found after dropping would be noted.

1. Weight (quantitative)
    a. System weight: 0.68lbs
    b. Simulated component Weight: 1.05g

2. Height (quantitative)
    a. Drop height: 30 feet
3. Damage (qualitative)
    a. Visual inspection and interpretation of damage

Methods

1. Attach the frame to the end of a fishing pole for retrieval.
2. Put the desired weight onto the frame to simulate the weight of the entire system.
3. Drop the system off of the roof and record the drop via video camera.
4. Retrieve the system and record any damage
5. Repeat the wanted for the desired number of trails

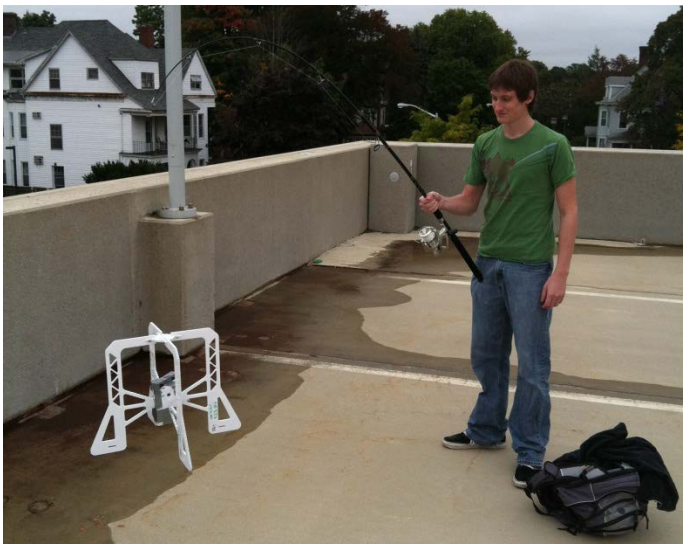


Figure 104: Chassis dropping location

Results

1. The frame of the robot held together through all three drops of the robot.
2. During the first drop, the robot flipped in the air and landed upside-down. No damage was sustained.
3. For the second drop, the frame flipped and landed upside-down. There was a crack on one of the sides of the frame
4. On the third drop, the extra weight that was attached to the frame was moved to the bottom of the electronics box. On this test the frame fell down without flipping. No noticeable damage was taken from the fall.

Figure 105: Damage sustained from the drop test

This test provided excellent evidence that corrugated plastic was a proper choice of material for IPASS's frame. The damage to one of the wings was in a weak location due to a series of holes designed to reduce weight. The weight reduced due to these holes is a minimal amount and this part of the chassis can be easily strengthened.

Appendix E: GPS Precision Test

In order to evaluate the capabilities and accuracy of the GPS unit the team performed three tests. Each test consisted of a team member moving in a zig-zag pattern across the WPI football field with the GPS unit while it was recording data. This path was tracked three times with the team member walking slowly in the first test, jogging in the second test, and sprinting in the third test. Each unique latitude/longitude point recorded by the GPS was then plotted using Google Maps to indicate the accuracy of the GPS.

Results

When the team member was walking slowly, the GPS had trouble tracking his movements as seen in Figure 111: GPS tracking when walking slowly.

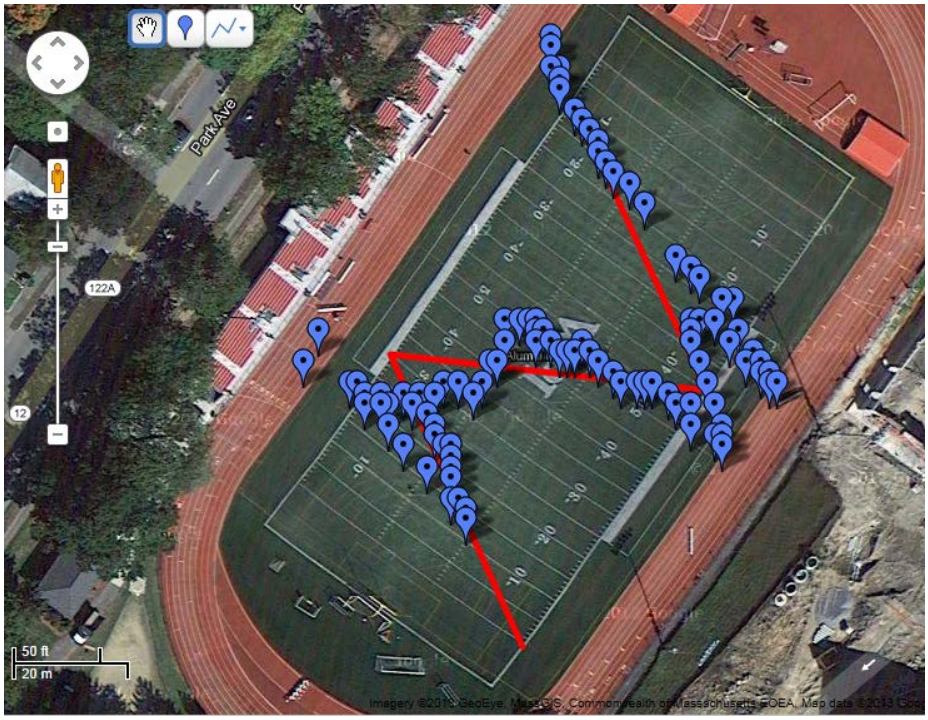


Figure 106: GPS tracking when walking slowly

While the team member was jogging and sprinting, the GPS was able to adequately track his position as seen in Figure 112 and Figure 113.

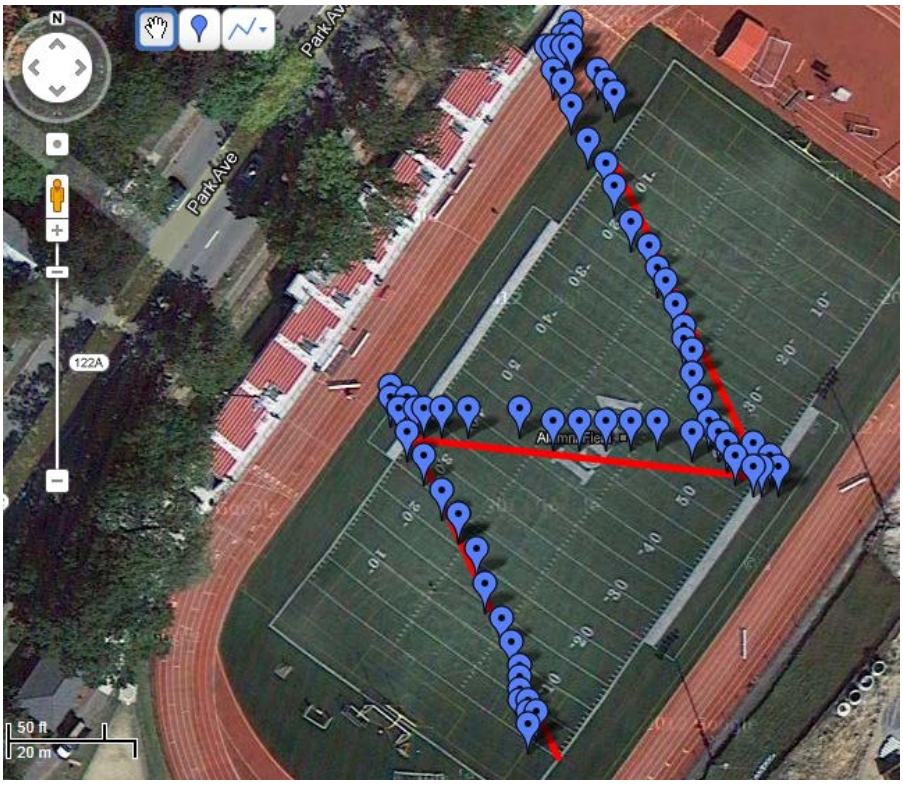


Figure 107: GPS tracking when jogging

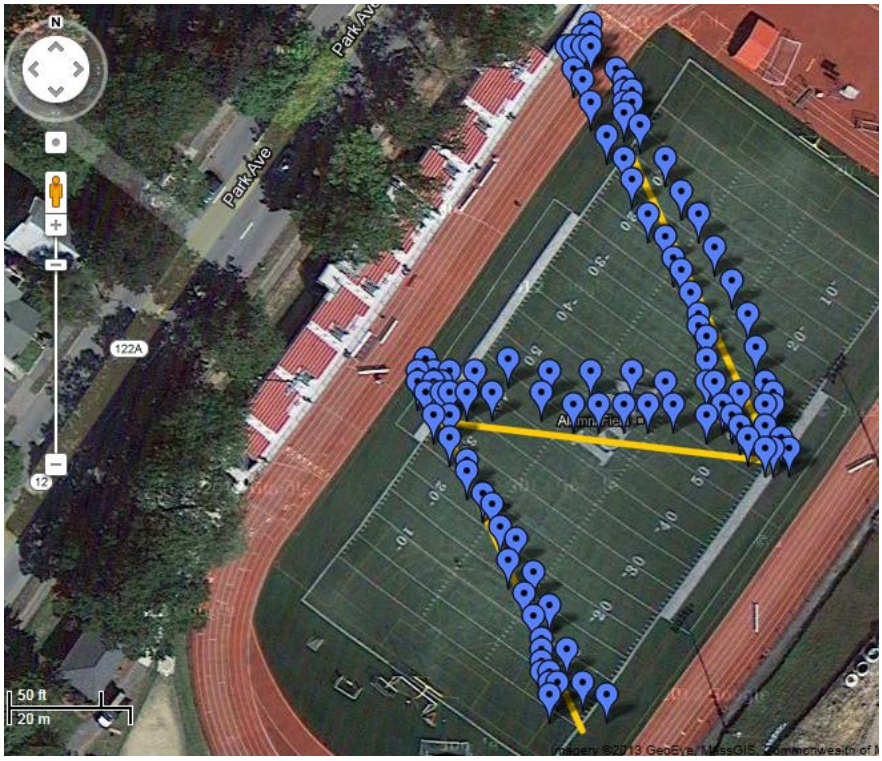


Figure 108: GPS tracking when sprinting

While the team member was sprinting, the GPS became less accurate but was still able to adequately track his movements. These three tests adequately evaluated the accuracy of the GPS. In its worst conditions (when the team member was walking) the GPS was off by a length of six meters. At its best condition (when the team member was jogging) the GPS was off by a length of one meter. While the height of the GPS did not vary wildly in this test, by tracking the deviations in altitude compared to the actual height it was carried at it was determined the GPS can track elevation to an accuracy of one half meters.

Appendix F: Electronics Box test

The camera setup in the electronics box was tested multiple times in a variety of environmental conditions to evaluate its capabilities. To simulate altitude from flight, the electronics box was pointed out the third and fourth story windows of buildings on WPI's campus. To simulate motion that would occur while airborne, the electronics box was also carried by a team member and walked around campus. The electronics box was also pointed out a window to capture image data at night to evaluate its capabilities with low or no light

Figure 109: Three unstitched images captured by each camera.

Figure 110: Resultant stitched image.

Figure 111: Stitched Image data captured from two stories (about 20 ft.) up.

Figure 112: Stitched image data captured from three stories (about 30 ft.) up.

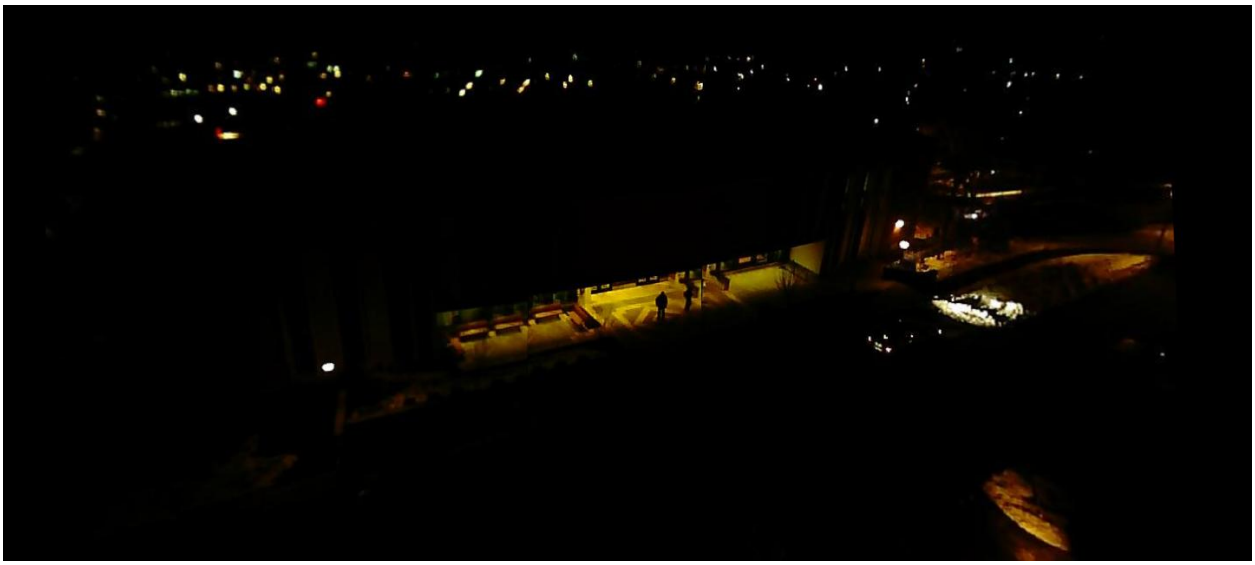


Figure 113: Stitched image data captured from the same location at night.

While the electronics box was in motion, it was common for there to be some amount of redundancy and distortion in the images. Figure 119 shows a scenario where both the electronics box and the subjects being photographed are moving independently of each other.

Figure 114: Stitched image data captured while the electronics box was in motion

To simulate IPASS operation, the electronics box was attached to a fishing line and lowered out of a fourth story window. One team member pulled the electronics box up allowing for the ground station to receive image data while the electronics box was raised. One stitched

image from this simulation can be seen in

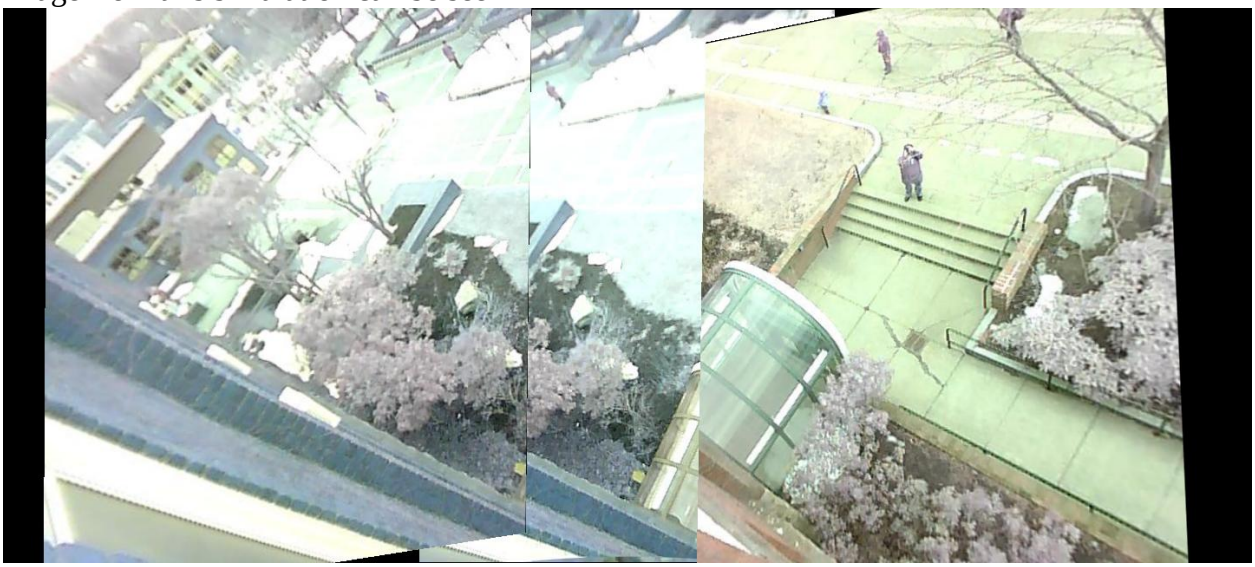


Figure 120.The repeated images seen on the left and middle images are resultant of electronics box motion.

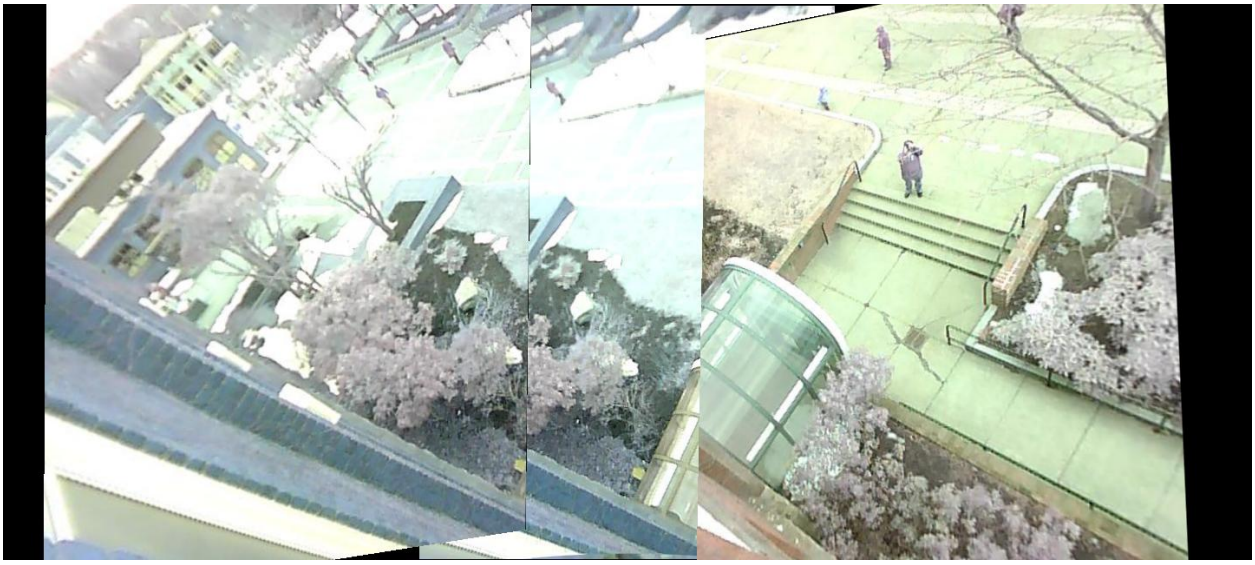


Figure 115: Stitched image from simulation of IPASS operation.
Notice that people are clearly visible in the image.

While image stitching functioned satisfactorily, two errors occurred in image capture during testing. The first was that individual cameras would sometimes produce blue tinted images as seen in Figure 121. This blue tint was attributed to the auto-gain of the camera model that was exacerbated by uneven lightning conditions and thus was significantly reduced outside. The second error was when no image data was received from a camera as seen in Figure 122. This error was caused by loose connection in the camera's board and was solved by bypassing the pin connection and soldering wires directly to the camera.

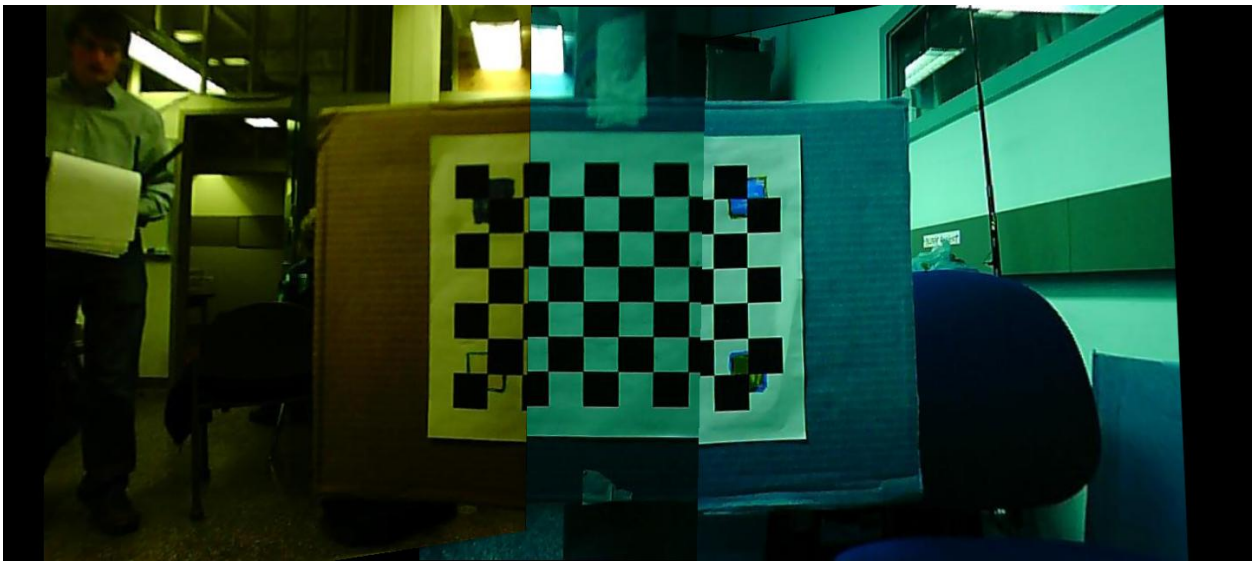


Figure 116: Stitched image data in which two of the cameras produced blue tinted images.

Figure 117: Stitched image data in which the leftmost camera was disconnected.

Appendix G: Matlab Simulation of IPASS Launch

To assist the team in diagnosing launch issues, a graduate student working on a similar project developed a Matlab simulation of the IPASS that takes into account the forces applied by the propellers and the mass of the system. The simulation developed proved that in an ideal system where the center of mass is directly below and coaxial with the forces applied by the thrust of the propellers, the chassis would launch vertically as shown below in Figure 123.

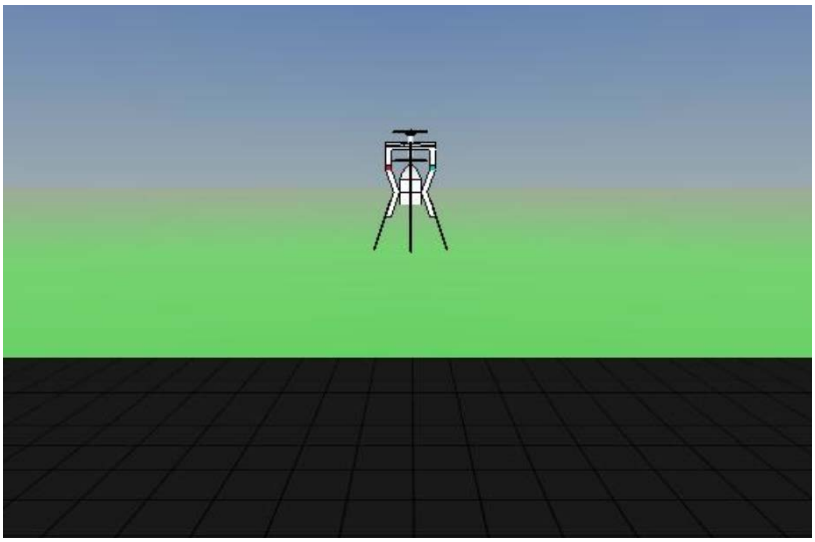


Figure 118: The IPASS launching in an ideal model

```
%------------------------ simulation control --------------------------%
% Simulation for Ipass
% Ruixiang 01/27/2013
%----------------------------------------------------------------------%

%clear
clear;
close all;
clc;

%Desired system state
height=50;
yaw=45/180*pi;

%Solve the ordinary differential equation for x
totalTime=500;
tspan=0:1:totalTime;
x0=zeros(4,1);
[t,x]=ode45(@(t,x) ipass(t,x,height,yaw),tspan,x0);
%[t,x]=ode113(@(t,x) quadrotor(t,x,flightMode,totalTime),tspan,x0);

%---------------------------- Animation ------------------------------%
% ATTENTION: If you don't have 3D Animation Toolbox installed in
```

```
% Matlab, you need to disable this part of code to run the simulation

%Rotation Matrix
% --phi----rotation angle about x axis
% --theta--rotation angle about y axis
% --psi----rotation angle about z axis
syms phi theta psi

Rxyz=[cos(psi)*cos(theta) -sin(psi)*cos(phi)+cos(psi)*sin(theta)*sin(phi)
sin(psi)*sin(phi)+cos(psi)*sin(theta)*cos(phi);
   sin(psi)*cos(theta) cos(psi)*cos(phi)+sin(psi)*sin(theta)*sin(phi) -
cos(psi)*sin(phi)+sin(psi)*sin(theta)*cos(phi);
   -sin(theta) cos(theta)*sin(phi) cos(theta)*cos(phi)];

world=vrworld('aircraft.wrl');
open(world);
fig=vrfigure(world);

xsize=size(x);
N=xsize(1);
set(world,'RecordMode','scheduled');
set(world,'RecordInterval',[1,N]);
set(fig,'Record2DFileName','ipass_ctrl.avi');
set(fig,'Record2D','on');
set(fig,'Record2DCompressQuality',100);
set(fig,'NavPanel','none');

for t=1:1:15
   world.aircraft.translation=[0 x(t,1) 0];
   %world.aircraft.rotation=[0 1 0 x(t,3)];
   world.aircraft.rotation=vrrotmat2vec(subs(Rxyz,[phi,theta,psi],[0-pi/2,-x(t,3),0]));
   set(world,'Time',t);
   vrdrawnow;
   pause(1);
end

close(world);
```

```
%---------------------------- Draw figure ------------------------------%
figure(1)
%plot(x(:,3));
plot(x(:,1),'r');
hold on
plot(x(:,3),'g');
hold off
legend('z','theta');
```

Figure 119: MATLAB simulation for IPASS flight in one dimension

During launch, the team discovered that the behavior exhibited by the actual chassis was dissimilar to the ideal simulation. The IPASS would launch vertically, tilt, and then crash. This simulation helped determine that the likely cause of this occurrence was due to an imbalance where the center of mass was not collinear with the force vector generated by the motors. This was the cause of the tilt that made the system crash. This behavior was also simulated and can be shown in Figure 125.

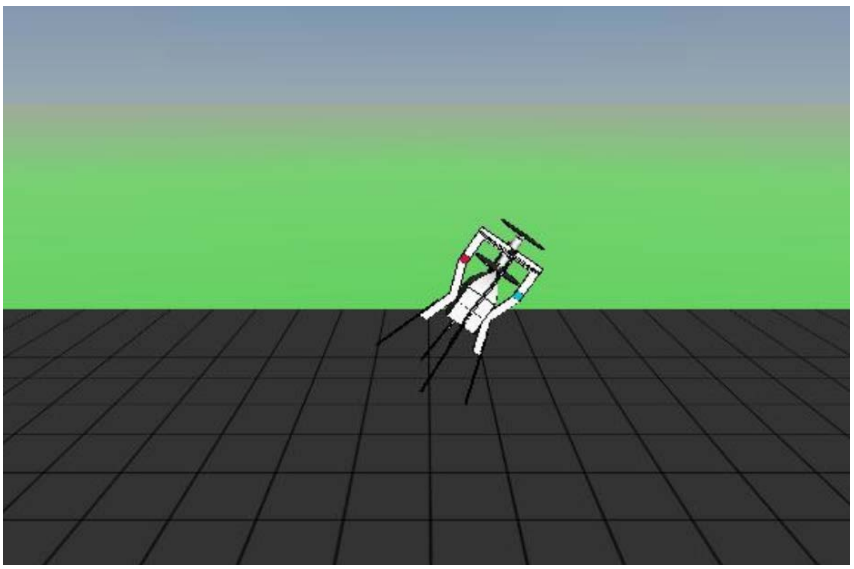


Figure 120: Model of the tiling behavior displayed by the IPASS

```
%------------------------ simulation control --------------------------%
% Simulation for Ipass
% Ruixiang 01/27/2013
%----------------------------------------------------------------------%

%clear
clear;
close all;
clc;
```

```
%Desired system state
height=50;
yaw=45/180*pi;

%Solve the ordinary differential equation for x
totalTime=10;
tspan=0:0.1:totalTime;
x0=zeros(6,1);
[t,x]=ode45(@(t,x) ipass(t,x,height,yaw),tspan,x0);
%[t,x]=ode113(@(t,x) quadrotor(t,x,flightMode,totalTime),tspan,x0);

%------------------------------ Animation ------------------------------%
% ATTENTION: If you don't have 3D Animation Toolbox installed in
% Matlab, you need to disable this part of code to run the simulation

%Rotation Matrix
% --phi----rotation angle about x axis
% --theta--rotation angle about y axis
% --psi----rotation angle about z axis
syms phi theta psi

Rxyz=[cos(psi)*cos(theta) -sin(psi)*cos(phi)+cos(psi)*sin(theta)*sin(phi)
sin(psi)*sin(phi)+cos(psi)*sin(theta)*cos(phi);
   sin(psi)*cos(theta) cos(psi)*cos(phi)+sin(psi)*sin(theta)*sin(phi) -
cos(psi)*sin(phi)+sin(psi)*sin(theta)*cos(phi);
   -sin(theta) cos(theta)*sin(phi) cos(theta)*cos(phi)];

world=vrworld('aircraft_2d.wrl');
open(world);
fig=vrfigure(world);

xsize=size(x);
N=xsize(1);
set(world,'RecordMode','scheduled');
set(world,'RecordInterval',[1,N]);
set(fig,'Record2DFileName','ipass_error.avi');
set(fig,'Record2D','on');
set(fig,'Record2DCompressQuality',100);
set(fig,'NavPanel','none');

T=600;        %Generate T animation
if N/T<1
   step=1;
else
   step=N/T;
end
```

```
index=1;

for t=1:step:N
   world.aircraft.translation=[x(index,6) x(index,5) 0];
   %world.aircraft.rotation=[0 1 0 x(t,3)];
   %world.aircraft.rotation=vrrotmat2vec(subs(Rxyz,[phi,theta,psi],[0-pi/2,-x(t,3),0]));
   world.aircraft.rotation=vrrotmat2vec(subs(Rxyz,[phi,theta,psi],[0-pi/2,0,-x(index,1)]));
   index=round(index+step);
   if index>N
      index=N;
   end
   set(world,'Time',t);
   vrdrawnow;
   %pause(1);
   if x(index,5)<0
      break;
   end
end

%close(world);

%---------------------------- Draw figure ------------------------------%
figure(1)
%plot(x(:,3));
plot(x(:,1),'r');
hold on
plot(x(:,3),'y');
plot(x(:,4),'g');
plot(x(:,5),'c');
plot(x(:,6),'b');
hold off
legend('alpha','vz_robot','vx_robot','z_world','x_world');
```

Figure 121: MATLAB simulation for IPASS flight in two dimensions

The simulation suggested that if the chassis applied a rotation to the system it would increase stability. Angular momentum would counteract the tilt caused by the mass imbalance. This works similar to the way a football travels when thrown in a spiral. When the rotation is induced in the simulation, the trajectory that the IPASS follows is much closer to the desired vertical translation as shown in Figure 127. This simulation provided the team with additional insight into how to stabilize the launch of the system.

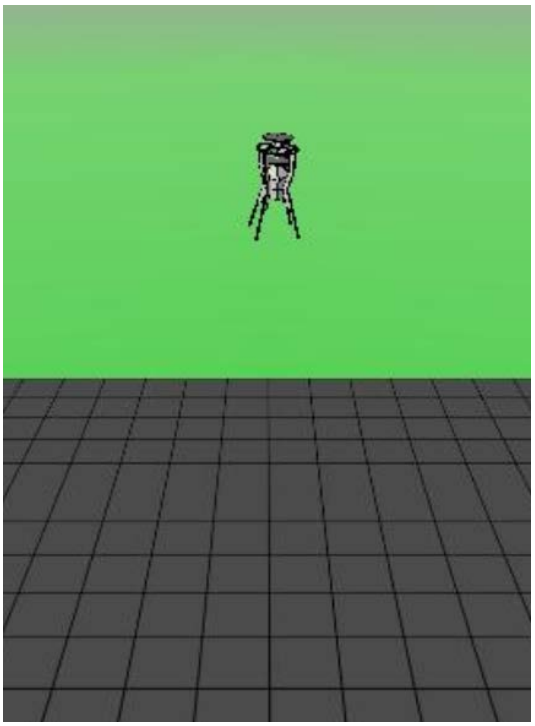


Figure 122: Model IPASS flight with rotations induced

```
%------------------------ simulation control --------------------------%
% Simulation for Ipass
% Ruixiang 01/27/2013
%----------------------------------------------------------------------%

%clear
clear;
close all;
clc;

%Desired system state
height=15;
yaw=45/180*pi;

%Solve the ordinary differential equation for x
totalTime=10;
tspan=0:1:totalTime;
x0=zeros(12,1);
[t,x]=ode45(@(t,x) ipass(t,x,height,yaw),tspan,x0);
%[t,x]=ode113(@(t,x) quadrotor(t,x,flightMode,totalTime),tspan,x0);

%---------------------------- Animation ------------------------------%
% ATTENTION: If you don't have 3D Animation Toolbox installed in
% Matlab, you need to disable this part of code to run the simulation
```

```
%Rotation Matrix
% --phi----rotation angle about x axis
% --theta--rotation angle about y axis
% --psi----rotation angle about z axis
syms phi theta psi

Rxyz=[cos(psi)*cos(theta)-sin(phi)*sin(psi)*sin(theta) -cos(phi)*sin(psi)
cos(psi)*sin(theta)+cos(theta)*sin(phi)*sin(psi);
   cos(theta)*sin(psi)+cos(psi)*sin(phi)*sin(theta) cos(phi)*cos(psi) sin(psi)*sin(theta)-
cos(psi)*cos(theta)*sin(phi);
   -cos(phi)*sin(theta) sin(phi) cos(phi)*cos(theta)];

Rx=[1 0 0; 0 cos(-pi/2) -sin(-pi/2);0 sin(-pi/2) cos(-pi/2)];

world=vrworld('aircraft_3d_far.wrl');
open(world);
fig=vrfigure(world);

xsize=size(x);
N=xsize(1);
set(world,'RecordMode','scheduled');
set(world,'RecordInterval',[1,N]);
set(fig,'Record2DFileName','ipass_3d2.avi');
set(fig,'Record2D','on');
set(fig,'Record2DCompressQuality',100);
set(fig,'NavPanel','none');

T=600;        %Generate T animation
if N/T<1
   step=1;
else
   step=N/T;
end
index=1;

for t=1:step:N
   world.aircraft.translation=[x(index,1) x(index,3) x(index,2)];

world.aircraft.rotation=vrrotmat2vec(Rx*subs(Rxyz,[phi,theta,psi],[x(index,10),x(index,11),x(in
dex,12)]));
   index=round(index+step);
   if index>N
      index=N;
   end
   set(world,'Time',t);
   vrdrawnow;
```

```
    pause(0.5);
    if x(index,3)<0
        break;
    end
end

%close(world);

%---------------------------- Draw figure ------------------------------%
figure(1)
%plot(x(:,3));
plot(x(:,1),'r');
hold on
%plot(x(:,11),'y');
plot(x(:,2),'g');
plot(x(:,3),'c');
% plot(x(:,6),'b');
hold off
legend('x','y','z');

figure(2)
%plot(x(:,3));
plot(x(:,10),'r');
hold on
%plot(x(:,11),'y');
plot(x(:,11),'g');
plot(x(:,12),'c');
% plot(x(:,6),'b');
hold off
legend('Angle_x','Angle_y','Angle_z');

%Plot trajectory
figure(3)
plot3(x(:,1),x(:,2),x(:,3),'-b','LineWidth',2)
axis([-80 80 -80 80 0 100])
```

Figure 123: MATLAB simulation of IPASS flight in three dimensions

Appendix H: Message Protocol

Command Token	Packet Type
0x00	Acknowledge
0x01	Pictures sent to Ground Station
0x11	IMU data sent to Ground Station
0x13	GPS data sent to Ground Station
0xE0	Launch sent to IPASS
0xF0	Land sent to IPASS
0xFF	Abort sent to IPASS

Acknowledge

Value	0x00	ID	Length	Previous CmdToken	Check Sum
Size	1 byte	1 byte	2 bytes	1 byte	1 byte

Pictures Sent

Value	0x01	ID	Length	Number Sent	Check Sum
Size	1 byte	1 byte	2 bytes	1 byte	1 byte

IMU Data

Value	0x11	ID	Length	String of , separated values	Check Sum
Size	1 byte	1 byte	2 bytes	N bytes	1 byte

GPS Data

Value	0x13	ID	Length	String of , separated values	Check Sum
Size	1 byte	1 byte	2 bytes	N bytes	1 byte

Launch

Value	0x0E	ID	Check Sum
Size	1 byte	1 byte	1 byte

Land

Value	0x0D	ID	Check Sum
Size	1 byte	1 byte	1 byte

Abort

Value	0x0F	ID	Check Sum
Size	1 byte	1 byte	1 byte

Appendix I: Ad-hoc Network Test

The strength of the ad-hoc network generated by the ground station was tested to determine its range. For this test, the electronics box remained stationary while a team member holding the ground station backed away. Images were received normally to a range of 75 ft. Beyond 75ft, frame rate dropped to one image every 20-30 seconds (down from one image every 3-5 seconds). At a distance of 120ft, the connection became incapable of sending images, and no more images were received by the ground station. This test showed that the ad-hoc wireless solution was sufficient for the current system's maximum height, but it did not meet the design specifications.

Appendix J: IPASS Flight Tests

Thrust Tests

During development of the IPASS chassis, multiple thrust and flight tests were conducted. While development was occurring on the first two chassis revisions, the thrust of the motor-propeller pairs were evaluated. Figure 129: Thrust test setup demonstrates the testing apparatus for early revisions of IPASS.

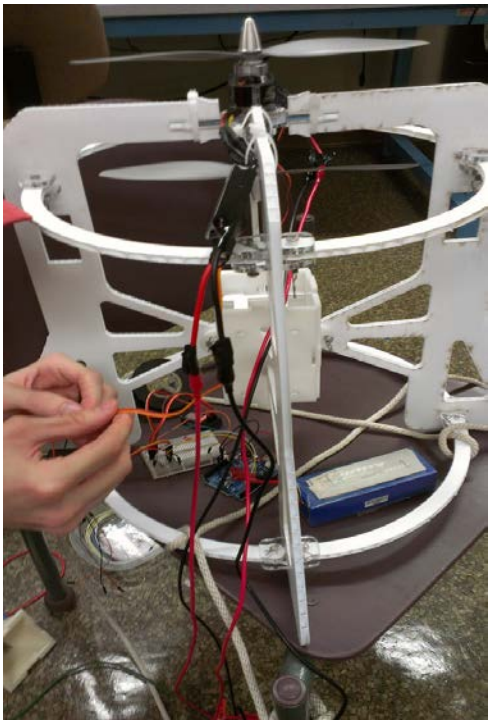


Figure 124: Thrust test setup

This thrust testing established that the original choice for motors were insufficient to launch the IPASS, and influenced the selection for their replacements. Similar testing was conducted with the newer, stronger motors and it was determined that sufficient thrust was achieved.

Launch Tests

Once the thrust of the IPASS propulsion system was determined sufficient for takeoff, launch tests were conducted. Following the procedures outlined in Appendix A: Safety Manual of IPASS, the team proceeded to attempt launch with multiple iterations of the chassis. Inconclusive tests were not included in this appendix.

November 27, 2012

This test was conducted underneath the Institute Park gazebo. This test featured the chassis design shown in Figure 37: Carbon fiber springs added for shock absorption and a string to measure height. The test demonstrated that the system was imbalanced, and simply tipped over without leaving the ground. Figure 130: First launch test demonstrates the tipping motion exhibited by this revision of the chassis.

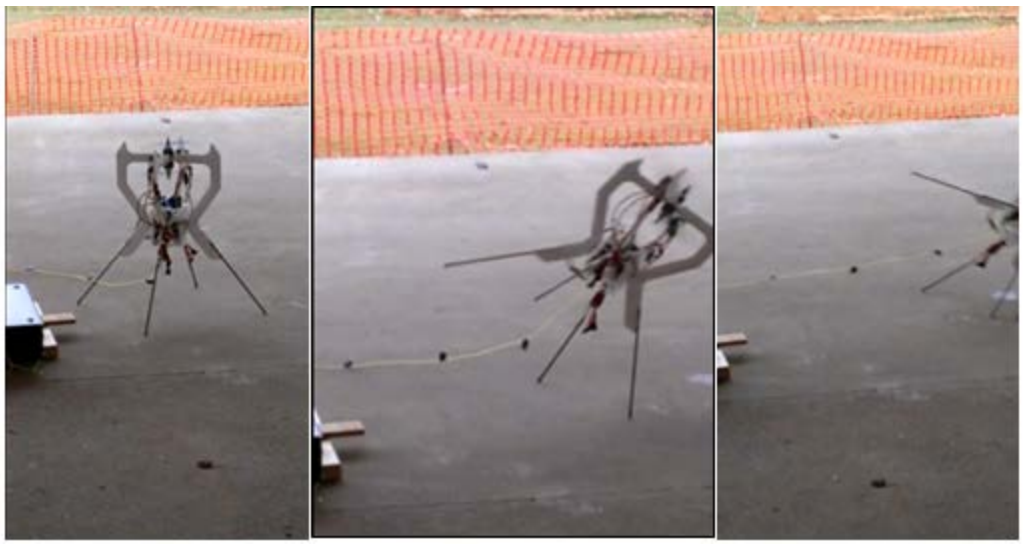


Figure 125: First launch test

January 12, 2013

This test was conducted under the Institute Park gazebo. The second flight test featured the design of the chassis shown in Figure 37: Carbon fiber springs added for shock absorption, and the team experimented with tethering the chassis until the motors reached maximum speed. An old Desktop computer was used as an anchor, and a rope was threaded through a tab to create the tether. As the motors reached maximum speed, the rope was released allowing the IPASS to leave the ground. This test also tipped sideways, never leaving the ground. This tipping motion is shown in Figure 131: Second launch test.

Figure 126: Second launch test

January 19, 2013

This test was conducted underneath the Institute Park gazebo. This test featured the same design described in the previous two tests and used a two-stage tether. This two-stage system featured a latch mechanism and a tether to establish a maximum height. When this test was

conducted, the latch mechanism allowed the motors to reach maximum speed before takeoff. When the latch was released, the IPASS immediately left the ground and reached the end of the tether. Once the tether became taught, the system turned horizontally and crashed to the ground. This motion is seen in Figure 132: Third launch test.

Figure 127: Third launch test

February 13, 2013

This test was conducted in Institute Park. This test featured the final revision of the chassis shown in Figure 16. For this test, the team chose not to use a tether, as the use of a tether would invalidate the project's success. This particular launch featured the same imbalance shown in previous tests, and the IPASS quickly turned horizontally and crashed. This result is shown in Figure 133: Fourth flight test.

Figure 128: Fourth flight test

On this day, a second test was conducted as well. For this second test, the propellers were switched to be co-rotating instead of contra-rotating. This was done to induce a rotation in the

IPASS while airborne, and increase stability. When the test was conducted, the system left the ground and featured the most stable operation of any test previous. After a short time, the rotation accelerated until stability was lost and the IPASS crashed. The system is shown during this test in Figure 134: Fifth flight test.

Figure 129: Fifth flight test

February 28, 2013

This test was conducted at night on the quad of WPI's campus. During flight, the IPASS launched about five feet into the air, hovered temporarily, and then crashed. One stitched image was captured during this flight. The image is of the WPI Recreational Center and can be seen in Figure 135.

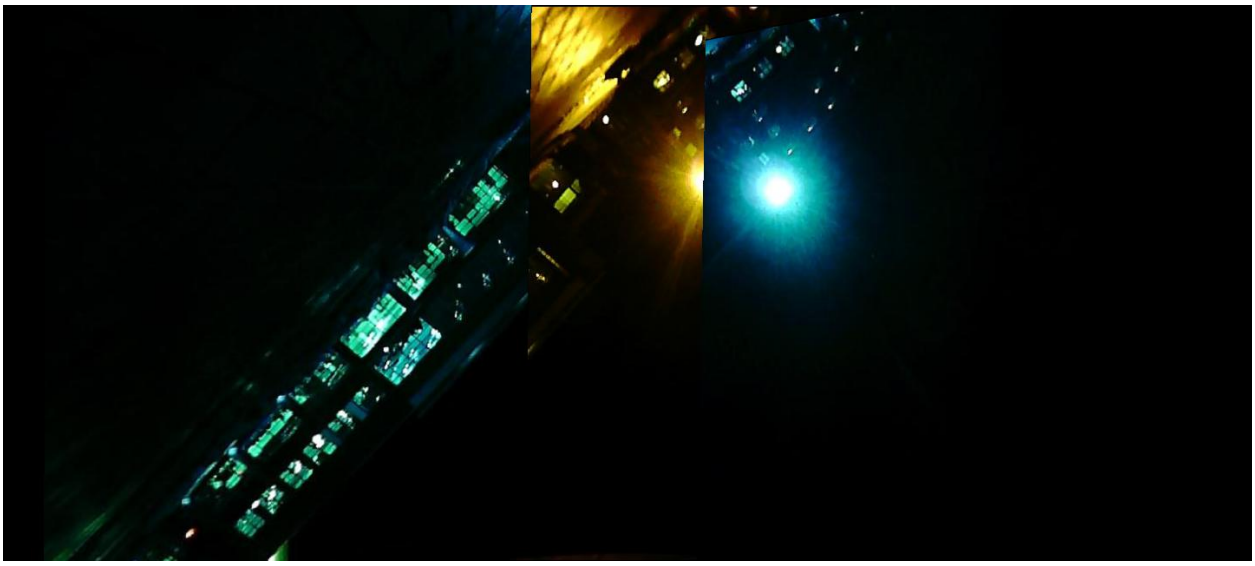


Figure 130: Image of the WPI Recreational Center captured while airborne

March 1, 2013

The final launch test was conducted on WPI's football field. During the test, wind gusts effectively prevented any launching of the IPASS. The team attempted to launch during periods of time with little wind, but no launch was achieved. Strong winds caused the system to tip over and never leave ground, as shown in Figure 136**:** Final flight test.

Figure 131: Final flight test

Appendix K: IPASS Survivability Test

The IPASS was dropped from a height of approximately 30 ft. in two consecutive instances. In the first drop test the IPASS sustained minimal damage and its electronics remained fully operational. Damage sustained from this drop test can be seen in Figure 137 and Figure 138.

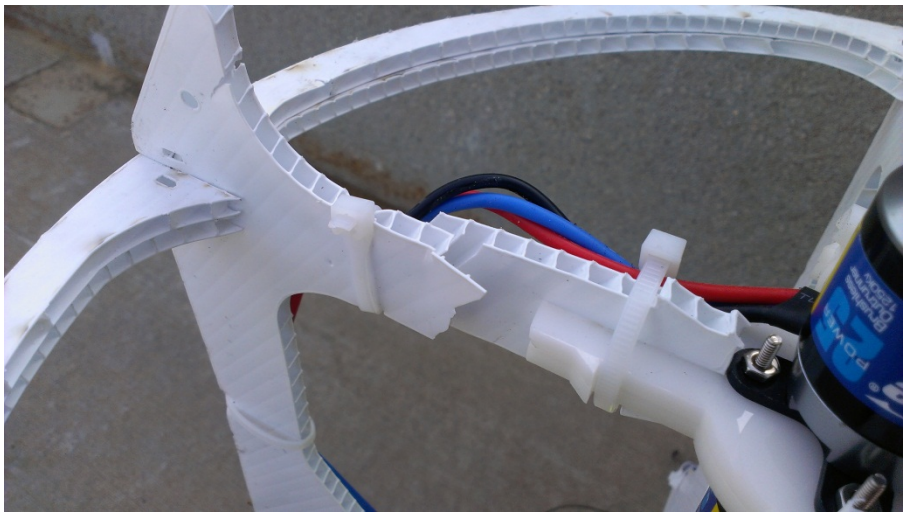


Figure 132: Damage sustained between the motor mount and propeller protection ring

Figure 133: Damage to the propeller protection ring

The second drop test was conducted immediately following the first test. Minimal repairs were made to the IPASS between tests involving resetting displaced Coroplast parts. No adjustments were made to the electronics box. Connection to the IPASS was lost after the second drop.

Figure 134: Landing site of the IPASS

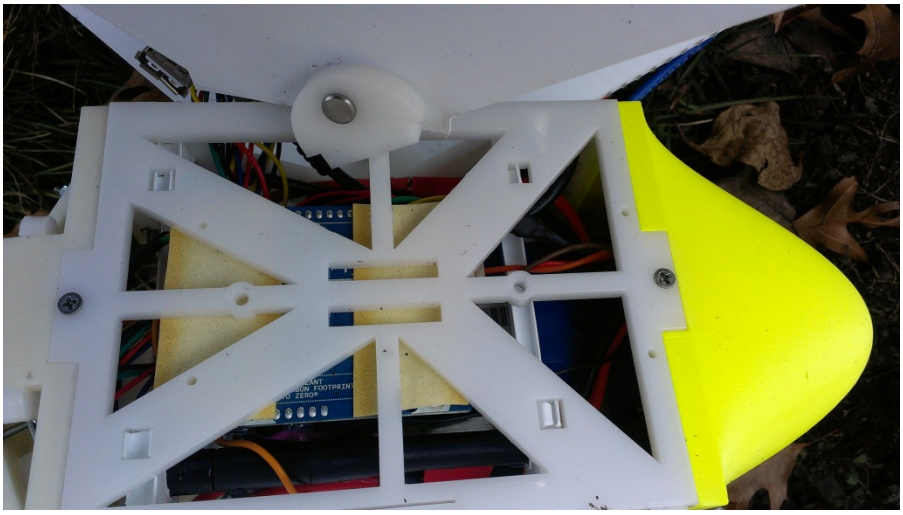


Figure 135: Close up of where the Delrin tabs came disconnected

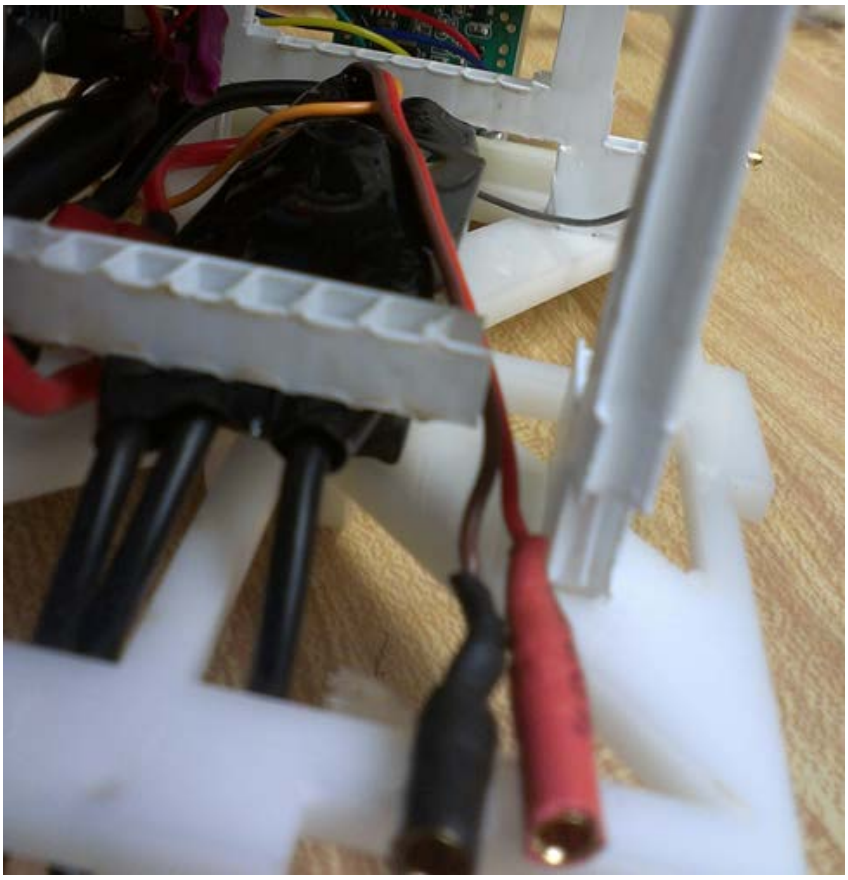


Figure 136: Broken battery mount inside the electronics box

The IPASS was taken apart after recovery to diagnose the damage caused during the tests. It was discovered that the Gumstix had become unseated from the Tobi board. Additionally the antennae and the SD had become dislodged from their mounts on the Gumstix.

As can be seen from these results, it has been shown that the IPASS can survive a 30 ft. free fall with little to no damage to the key electronics components

Appendix L: Lenovo Thinkpad Specifications

Table 12: Lenovo Thinkpad W530 specifications

System Components
Intel Core i7-3630QM Processor (6M Cache, up to 3.40 GHz)
Windows 7 Professional (64 bit)
Windows 7 XP Mode - English
15.6" HD+ (1600 x 900) LED Backlit AntiGlare Display, Mobile Broadband Ready
NVIDIA Quadro K1000M Graphics with 2GB DDR3 Memory
8 GB DDR3 - 1600MHz (2 DIMM)
Keyboard Backlit - US English
UltraNav with Fingerprint Reader
720p HD Camera with Microphone
128GB Solid State Drive, SATA3
DVD Recordable
Express Card Slot & 4-in-1 Card Reader
9 Cell Li-Ion TWL 70++
170W Slim AC Adapter - US (2pin)
Bluetooth 4.0 with Antenna
Intel Centrino Advanced-N 6205 AGN
Mobile Broadband upgradable

Appendix M: IPASS Cost Breakdown

Table 13: Cost of each component to construct an IPASS

Component	Quantity	Unit Cost	Total Cost
18" x 24" Coroplast Sheet	3	$ 25.92	$ 77.76
18" x 24" Delrin Sheet	1	$ 39.87	$ 39.87
Gumstix Overo FE COM	1	$ 229.00	$ 229.00
Tobi Expansion Board	1	$ 69.00	$ 69.00
Arduino Pro Micro	1	$ 19.95	$ 19.95
Jameco Prototyping Board	1	$ 6.00	$ 6.00
Eflite Power 25 Motor	2	$ 69.95	$ 139.90
SB101C USB CMOS Modules	3	$ 31.49	$ 94.47
Zippy Flightmax LifePo4 Battery	1	$ 43.21	$ 43.21
APC 10" x 4.7" Propeller Pair	1	$ 9.59	$ 9.59
Clevis Pins	4	$ 1.41	$ 5.64
Zip Ties	18	$ 0.01	$ 0.18
4-40 x 5 Screws	8	$ 0.65	$ 5.20
4/40 Nuts	8	$ 0.74	$ 5.92
Camera Mounting Screw	12	$ 0.42	$ 5.04
Nose Cone	1	$ 38.46	$ 38.46
Camera Mount	1	$ 81.92	$ 81.92
Total			**$ 871.11**

Appendix N: Full Expense Report

Table 14: Total Expenses for thes IPASS Project

8/18/2012	Sparkfun - Arduino	$ 23.54	$ 7,976.46
	Mouser - electronics	$ 15.73	$ 7,960.73
	Adafruit - Raspberry pi accessories	$ 15.55	$ 7,945.18
	Newegg - electronics	$ 16.48	$ 7,928.70
9/21/2012	Pack n Seal - Coroplast	$ 25.92	$ 7,902.78
9/26/2012	HobbyKing - motors	$ 87.16	$ 7,815.62
	Amazon - electronics	$ 19.18	$ 7,796.44
	Robotshop - IMU	$ 91.27	$ 7,705.17
9/28/2012	Pack n Seal - Coroplast	$ 46.56	$ 7,658.61
10/2/2012	Digi - Xbee wifi	$ 42.89	$ 7,615.72
	Electronics 123 - SPI cameras	$180.43	$ 7,435.29
	Amazon - GPS	$ 32.17	$ 7,403.12
10/5/2012	Sparkfun - Arduino	$ 28.59	$ 7,374.53
10/30/2012	McMaster - mechanical parts	$ 27.21	$ 7,347.32
	Adafruit - Raspberry pi accessories	$ 34.63	$ 7,312.69
	Newegg - electronics	$ 9.99	$ 7,302.70
10/31/2012	McMaster - mechanical parts	$ 18.81	$ 7,283.89
11/1/2012	Amazon - electronics	$ 16.28	$ 7,267.61
11/2/2012	Rapid Prototype - nose cone	$ 41.68	$ 7,225.93
11/2/2012	McMaster - mechanical parts	$ 85.66	$ 7,140.27
11/5/2012	HobbyKing - motors	$ 84.70	$ 7,055.57
11/14/2012	McMaster - mechanical parts	$ 55.00	$ 7,000.57
11/19/2012	New Egg - electronics	$ 17.98	$ 6,982.59
	Amazon - electronics	$ 26.64	$ 6,955.95
11/27/2012	Pack and Seal - Coroplast	$ 87.59	$ 6,868.36
11/28/2012	Zachary Mintz - motors & esc	$ 91.84	$ 6,776.52
	Adam Blumenau - motors	$ 74.36	$ 6,702.16
	Adam Blumenau - propellers	$ 12.89	$ 6,689.27
	Alec Ishak - propellers	$ 51.60	$ 6,637.67
11/30/2012	McMaster - mechanical parts	$ 13.07	$ 6,624.60
12/3/2012	McMaster - mechanical parts	$153.17	$ 6,471.43
	Rapid Prototye - camera mount	$ 81.92	$ 6,389.51
12/4/2012	Amazon - anemometer	$ 38.98	$ 6,350.53
	Returned - anemometer	$(38.98)	$ 6,389.51

12/4/2012	Hobbyking - battery	$ 43.15	$ 6,346.36
12/13/2012	Gumstix	$ 369.98	$ 5,976.38
1/8/2013	Gumstix	$ 84.95	$ 5,891.43
1/14/2013	Amazon - mechanical parts	$106.31	$ 5,785.12
1/16/2013	New Egg - electronics	$ 13.99	$ 5,771.13
1/22/2013	Numato - level shifters	$ 34.49	$ 5,736.64
	Mouser - level shifters	$ 20.39	$ 5,716.25
	Gumstix	$ 28.24	$ 5,688.01
	Rapid Prototype - WSU part	$ 2.36	$ 5,685.65
1/23/2013	Alec Ishak - SD card	$ 19.11	$ 5,666.54
1/30/2013	Electronics123 - USB cameras	$112.85	$ 5,553.69
2/4/2013	Laptop	$1,878.00	$ 3,675.69
2/4/2013	Gumstix	$ 305.46	$ 3,370.23
	Sparkfun -	$ 68.64	$ 3,301.59
	New Egg - misc electronics	$ 18.79	$ 3,282.80
	Electronics123 - USB cameras	$112.85	$ 3,169.95
	Robotshop.com - IMU	$ 96.17	$ 3,073.78
	Amazon - mehcanical parts & GPS	$ 61.33	$ 3,012.45
	Amazon - mechanical parts	$ 234.96	$ 2,777.49
	Hobbyking (hextronix) - brushless motors	$ 96.29	$ 2,681.20
	Hobbyking - battery and ESC	$ 73.04	$ 2,608.16
	McMaster - mechanical parts	$ 82.97	$ 2,525.19
	Amazon - mechanical parts	$ 21.80	$ 2,503.39
	Dragonfly - propellers	$ 62.00	$ 2,441.39
	Pack and Seal - Coroplast	$ 22.24	$ 2,419.15
2/12/2013	Corey Russell - motor	$ 74.36	$ 2,344.79
2/13/2013	Rapid Prototype - nose cone	$ 38.56	$ 2,306.23
2/21/2013	Pelican Case	$274.68	$ 2031.55

Appendix O: IPASS Instruction Manual

Intelligent Portable Aerial Surveillance System (IPASS) Assembly Instructions

IPASS Inventory

I. Electronics Box
   a. 4x Delrin box walls
   b. 8x Delrin electronics box mounts
   c. 1x Nose cone
   d. 1x Camera mount
   e. 1x Gumstix Overo FECOM
   f. 1x TOBI board
   g. 1x Arduino Pro Micro
   h. 1x RC receiver
   i. 1x Globalsat ND100s USB GPS receiver
   j. 2x Turnigy Trust 55A SBESC
   k. 1x battery Y connector
   l. 1x 5500 mAhr LiFePo battery
   m. 3x SP101c USB cameras
   n. 8x 4-40x5/8 screws
   o. 8x 4/40 nuts
   p. 8x camera screws

II. Chassis
   a. 2x E-flite brushless Outrunner Motor 1250 Kv
   b. 2x Propeller mounts
   c. 2x 10x4.7 Slo-Flyer composite propellors
   d. 4x 4/40 1” screws
   e. 4x 4/40 nuts
   f. 6x Motor lead extensions
   g. 2x Delrin motor mounts
   h. 4x Carbon fiber tubes
   i. 2x Coroplast rings
   j. 4x Coroplast supports
   k. 4x Clevis pins
   l. 24. Zip ties

III. Groundstation
   a. 1x Lenovo Thinkpad W530 laptop
   b. 1x Lenovo Thinkpad charger

Setup

Gumstix

There are four parts to the Gumstix system. The Overo FE COM, henceforth referred to as the Gumstix, the Tobi expansion board, the SD card, and the antennas. Only one of the two antennae is needed. Plug the antenna into the top antenna port when the Gumstix is held so that the text on the Gumstix is upright. The Gumstix can then be attached to the Tobi board.

To load the Gumstix Overo image you will need a microSD card with storage greater than 2 GB, a method of connecting the microSD card to your computer, and a Linux environment with administrative access.

1) Select the latest stable image from the Gumstix repository: http://cumulus.gumstix.org/images/angstrom/factory/
2) Download the omap3-desktop-image, u-boot.bin, MLO and uImage files.
3) Connect the microSD card to the computer
4) If your Linux environment automatically mounts the microSD card you will need to unmount it. Typically the microSD card appears under /dev/mmcblk0 or /dev/sdc. The command to unmount the microSD card is:

```
sudo umount –l /dev/mmcblk0
```

5) Now you will need to record the exact size of the microSD card in bytes. The size of the microSD card will be displayed in the first line of results of the following command:

```
sudo fdisk –l /dev/mmcblk0
Disk /dev/mmcblk0: 2016 MB, 2016411648 bytes
```

6) Now you will need to erase the boot partition of the microSD card, this will allow you to make a bootable microSD card so that the environment can be loaded on to the Gumstix. To erase the partition table run the following command:

```
sudo dd if=/dev/zero of=/dev/mmcblk0 bs=1024 count=1024
```

7) Now you will need to calculate the cylinder count of your microSD card. This value is only an approximation and is only used because the software required to create the partition table has been adapted from standard hard drives. To find the cylinder count you will need to divide the size of the microSD card (in bytes) by 255 heads, 63 sectors per head, and finally 512 bytes per sector. For example: this value will be 245.15, which when rounded down is 245 cylinders.
8) Take the value you calculated in step 7 and insert it into the highlighted section in the command below:

```
sudo sfdisk –force –D –uS –H 255 –S 63 –C 245 /dev/mmcblk0
```

9) The sfdisk command will ask you for some additional information. Please enter the highlighted information when prompted:

```
Checking that no-one is using this disk right now ...
OK
[snip]
Input in the following format; absent fields get a default value.

Usually you only need to specify  and  (and perhaps ).
/dev/mmcblk0p1 :128,130944,0x0C,*
/dev/mmcblk0p1   *   128    131071 130944   c  W95 FAT32 (LBA)
/dev/mmcblk0p2 :131072,,,-
/dev/mmcblk0p2    131072   3938303    3807232  83  Linux
/dev/mmcblk0p3 :
/dev/mmcblk0p3     0     -  0   0  Empty
```

```
/dev/mmcblk0p4 :
/dev/mmcblk0p4     0     -  0   0  Empty
New situation:
Units = sectors of 512 bytes, counting from 0
  Device Boot    Start   End   #sectors  Id  System
/dev/mmcblk0p1   *   128    131071 130944   c  W95 FAT32 (LBA)
/dev/mmcblk0p2    131072   3938303    3807232  83  Linux
/dev/mmcblk0p3     0     -  0   0  Empty
/dev/mmcblk0p4     0     -  0   0  Empty
Warning: partition 1 does not end at a cylinder boundary
Do you want to write this to disk? [ynq] y
Successfully wrote the new partition table
[snip]
```

10) Now that the microSD card has been partitioned you will need to write valid file systems to the two partitions.
11) To format the boot partition please enter the following command, please note the p1 at the end of the device as this signifies the first partition:

```
sudo mkfs.vfat –F 32 /dev/mmcblk0p1 –n boot
```

12) If the following commands fails and you are presented with an error message stating that your computer is missing “mkfs.vfat” you will need to install the “dosfsutils” package and try again.
13) Now you will need to format the storage partition. To do so please enter the following command, please note the p2 at the end of the device as this signifies the second partition:

```
sudo mke2fs –j –L rootfs /dev/mmcblk0p2
```

14) Now you will need to mount the partitions so that you can copy files. To do so enter the following commands:

```
sudo mkdir /media/{boot,rootfs}
sudo mount –t vfat /dev/mmcblk0p1 /media/boot
sudo mount –t ext3 /dev/mmcblk0p2 /media/rootfs
```

15) Now you will need to copy the downloaded data over to the microSD card. This can be accomplished with the following commands:

```
sudo cp MLO /media/boot/MLO
sudo cp u-boot.bin /media/boot/u-boot.bin
sudo cp uImage /media/boot/uImage
```

16) Finally you will need to unarchive the omap3-desktop-image folder onto the microSD card. This can be accomplished with the following two commands:

```
sudo tar xaf rootfs.tar.bz2 –C /media/rootfs
sync
```

17) The microSD card is now ready to be inserted into the Gumstix for the first time boot procedure.

Once the Linux image is loaded onto the SD card, insert the SD card into the Gumstix. Attach the Gumstix to the Tobi board and plug in the Tobi’s power cable. Make sure that there is a monitor, keyboard, and mouse attached to the Gumstix, a USB hub may be needed. After the Gumstix is done booting a few updates will need to be installed to get the Gumstix working with the IPASS. Make sure the Gumstix has access to the internet. In the terminal, type:

```
opkg update
opkg install task-native-sdk
```

This will update the Gumstix repositories and download the necessary packages needed to run IPASS software.

Copy the given IPASS file system into the root directory. This contains the scripts needed to run the cameras, GPS and C code needed for wireless communication.

Disconnect the Gumstix from the internet. While the ad-hoc network is being hosted on the ground station, click on the network icon on the Gumstix and then click on "Connect to Hidden Network." For complete instructions on how to set up the ad-hoc network on the ground station, please see the ground station setup section below. If a network was created, select the network in the Ad-Hoc Network section and enter the password. Similar to the ground station, go into the edit wireless network tool and change the Netmask and Gateway to the same values used in the ground station. Change the IP address to be 10.42.43.2. As long as the Gumstix is not connected to a wired network and has no other wireless networks remembered it will connect to the ad-hoc network as long as it is being hosted by the ground station.

To set up the image transfer capabilities the Gumstix needs to be able to access the ground station without entering a password. First, ssh from the Gumstix to the ground station laptop and ssh into the ground station laptop from the Gumstix. This will create the ssh folder in the root directory of both systems. Then enter the commands below in the terminal of the Gumstix. You will need to replace all instances of groundstation with the ground station user name and groundstationIP with the IP address of the ground station on the ad-hoc network.

```
ssh-keygen –t dsa
scp ~/.ssh/id_dsa.pub groundstation@grundstationIP:.ssh/authorized_keys2
ssh-agent sh –c 'ssh-add < /dev/null && bash'
```

Entering ssh groundstation@groundstationIP from the Gumstix will give access to the terminal on the ground station with no password prompt.

Arduino

As the Arduino Pro Micro is not directly manufactured by Arduino, the setup is different than described on the Arduino web page. To begin the process, the Arduino programming environment should be installed. This software can be easily downloaded at http://arduino.cc/en/main/software. The arduino environment is easy to use and install in windows, though required libraries are severely outdated for Mac OSX and Linux. Because of this, it is strongly suggested that the Arduino software be used only in Windows. The rest of this section will assume a Windows 7 environment.

Once the Arduino environment has been installed, the Arduino Pro Micro drivers must be downloaded from the Sparkfun website. These drivers can be found on the product page(https://www.sparkfun.com/products/11098?) , or directly at http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Dev/Arduino/Boards/SF32u4_boards.zip. The files contained within the .zip folder are to be placed in the Arduino folder on the hard drive. Once the files are merged, connect the device to your computer using the USB connection. As the connection is made, the device must be manually directed to use the Arduino drivers. More detailed instructions on this process can be found at http://www.sparkfun.com/tutorials/338.

After the arduino is properly recognized by the computer, the COM port used by the arduino must be discovered. This can be accomplished by opening “Devices and Printers” in the start menu and looking at the entry “Arduino Pro Micro”. After the COM port is established, the .ino file included with the IPASS can be opened in the Arduino programming environment. In the Arduino environment, select “tools” and “board”. Within this drop-down menu select the “Sparkfun Pro Micro 5v/16MHz”. After the board is selected, enter “tools” and “Serial Port”, selecting the COM port associated with the Arduino.

With the Arduino environment properly setup, the only remaining step is to click the “upload” button represented by a horizontal arrow on the top of the Arduino environment. Once this process is complete, the arduino is ready to be integrated into the system. `

Cameras

The three SB101C camera modules come with a connector with a USB B output. To connect to the Gumstix, cut the given USB B connection off of the cable. The individual wires can be soldered directly to a USB hub for easy attachment/detachment from the Gumstix. Alternatively, USB A connections can be soldered on so that each camera may be removed from the USB hub individually. However, this is not recommended because typical USB connections are meant to be removed and could come loose during operation. For best connectivity, solder the connection wires directly to both the cameras and the USB hub.

To ensure operation of the cameras, plug them into the Gumstix then, in the terminal, navigate to the /dev folder. If properly connected, the three cameras will appear in this folder as video0, video1, and video2.

Motors

Connect the ESCs to the Arduino Pro Micro. Power the Arduino and turn on the RC controller. The motors should play a short song to indicate that they are receiving power. Test the motor direction by pushing on the throttle of the RC motor slightly. If one or more motors are running in the incorrect direction, switch two motor signal cables to the ESCs to reverse the direction of the motor spin. If the motors are running in the correct direction, mark the signal cables and their corresponding plug for quicker setup in the future.

Ground Station

You will need to install Open SSH on the ground station to use the IPASS. In the terminal of the ground station, type:

```
sudo apt-get install OpenSSH
```

The Ad-hoc network is hosted from the ground station. To create this network, select the network options and click on *connect to a new wireless network* button. Enter the connection name and the SSID. It is easiest if these two fields are given the same name. Select the mode to be Ad-hoc. Under the wireless security tab, you may set a password if desired. Under the IPv4 Settings, select manual input. Add 10.42.43.1 as the address 255.255.255.0 as the Netmask, and 0.0.0.0 as the Gateway. Save these changes.

Open the network manager and select connect to a hidden wireless network. Select the name of the network created and select connect. The ground station will then start hosting the ad-hoc network.

In order to stitch the images coming from the IPASS the ground station will need to have OpenCV installed. To install OpenCV onto the ground station please follow these instructions:

1) Download the latest version of OpenCV for Linux from here: http://opencv.org/downloads.html
   The download can take several minutes as it is a large file.
2) Unarchive the OpenCV files. This process can take some time as OpenCV has a large unarchived size.
3) You will now need to install the “cmake” and “cmake-gui” packages on the ground station.
4) Create a new folder to store the compiled OpenCV binaries.
5) Open a new terminal window and navigate into the OpenCV source folder.
6) Open the cmake gui by running the following command:

```
sudo cmake-gui
```

7) In the cmake gui you will need to specify the source code folder and the built binaries folder. To specify the source code folder and click on the “Browse Source…” button and in the new window click on the “Open” button. To specify the built binaries folder click on the “Browse Build…” button and in the new windows navigate into the folder you created and then click on the “Open” button.
8) Now you will need to click the “Configure” button, in the next window make sure that “Unix Makefiles” and “Use default native compilers” are selected and then click the “Finish” buttton. This will generate the cmake script to build the OpenCV files for the ground station.

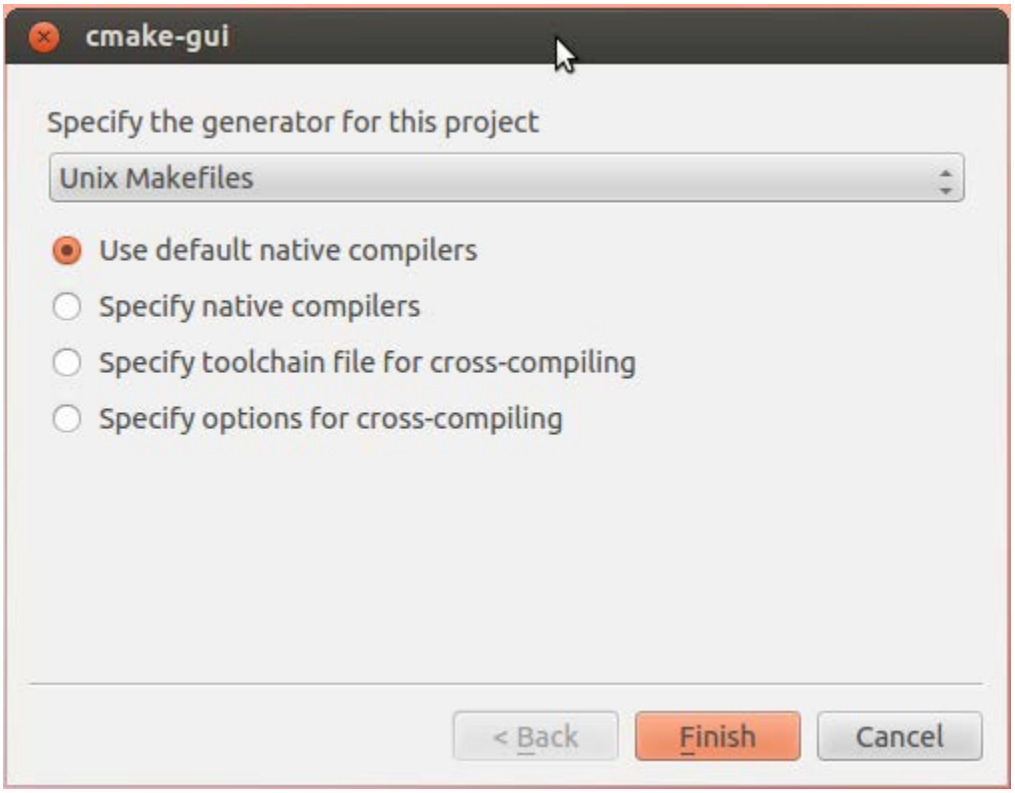


Figure 137: The cmake GUI build options

9) Once the configuration has finished the center area of the cmake gui will now display the possible build options for OpenCV, this will have a red background to signify that you have not selected any additional options. In order to properly operate the ground station you will need to select the following additional options:

```
BUILD_JPEG
BUILD_PNG
BUILD_TIFF
BUILD_ZLIB
```

Figure 138: The cmake GUI before adding additional build options

10) Now you will need to click the “Configure” button again to add these additional build options to the build script.
11) Once the configuration has finished the background will return to white and you will need to click the “Generate” button to create the build script.

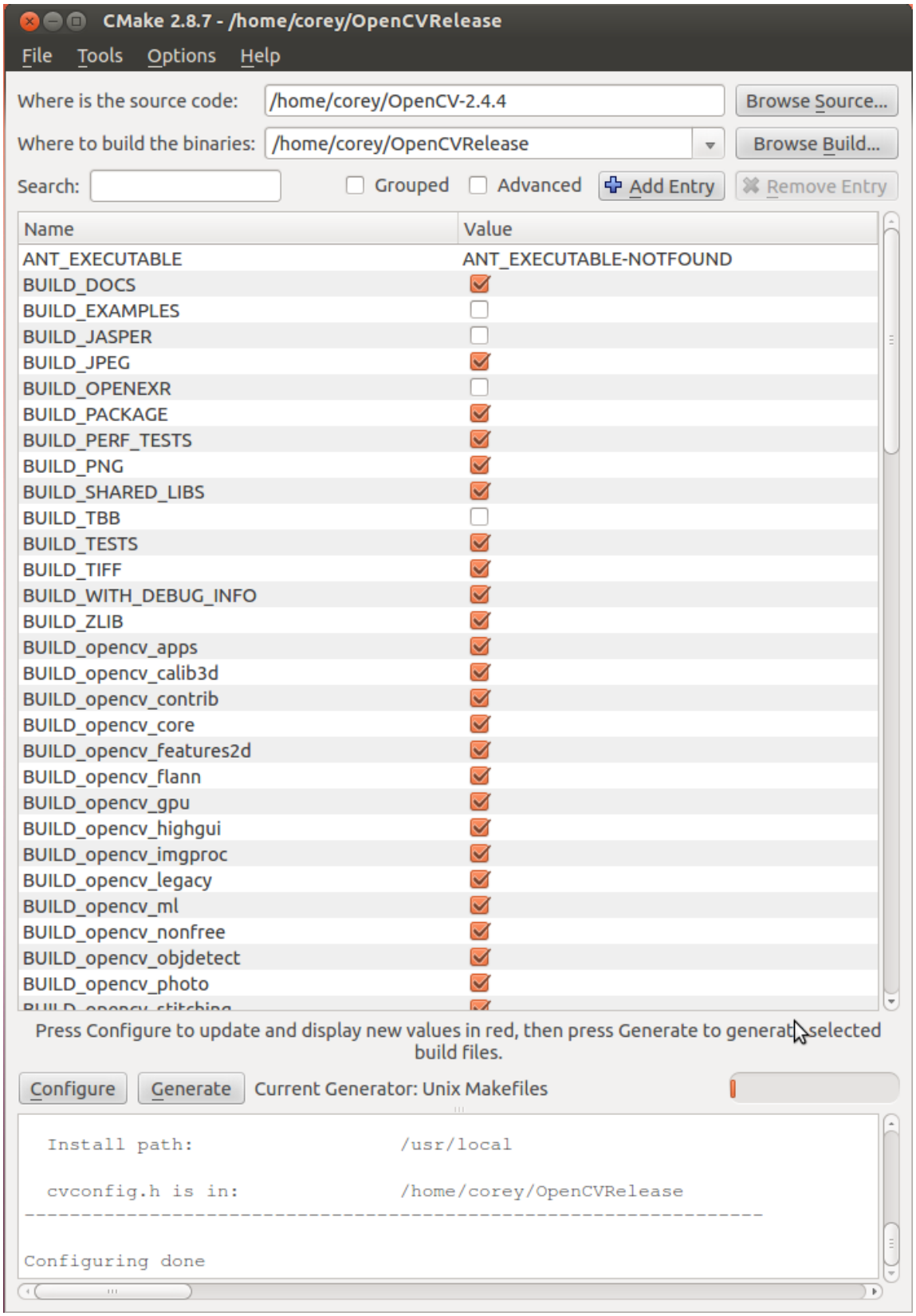


Figure 139: The cmake GUI after adding additional build options

12) Once the script has been generated you will need to close the cmake gui and return to the terminal window. Now you will need to navigate into the OpenCV binaries folder you created and run the “make” command. This run the script you generated in step 11 and then build OpenCV for the ground station. This process can take upwards of 1 hour to complete.
13) Once OpenCV has been built you will need to run the following command to install OpenCV onto the ground station.

```
sudo make install
```

14) Once this command finishes OpenCV will be fully installed and you will be able to run the IPASS GUI and stitch images properly.

Assembly

Motors

Mount both motors onto the two Delrin motor mounting plates. Mount propellers on each motor, making sure the text on both propellers is face up. To run contra rotating thrust generation, secure both propellers from the same pair onto the motor mounts with an acrylic spacer on each mount and connect the three motor signal cables to their corresponding ESC. To run co-rotating thrust generation, secure identical propellers onto the motor mounts with an acrylic spacer on each mount and connect the three motor signal cables to a corresponding ESC with two of the signal cables switched.

Landing Gear

The four carbon fiber tubes provided correspond to the four indents located on the bottom of the quarter panels. Secure a carbon fiber rod to a quarter panel as shown below.

Figure 140: Attaching a carbon fiber tube to a quarter panel

Electronics Box

To assemble the IPASS, as many connections as possible should be permanent. This can be achieved either through solder or hot glue. Use caution with working with both of these as any erroneous connections made during assembly can cause permanent damage.

Both the Arduino and Gumstix are powered from the 5V rails on the two ESCs. On the Gumstix, attach a wire from power to one of the ESC’s red wires and attach the ground from the Gumstix to the brown wire. On the Arduino, attach the other ESC’s red and brown wire to Vcc and one of the grounds respectively.

The wireless transmitter connects to the Arduino Pro Micro. Plug the included 3 wire connector into the transmitter so that the ground wire is closest to the outside of the transmitter. Connect the power, signal, and ground wires to Vcc, pin 3, and ground. The signal wires from the two ESCs are both connected to pin 9 on the Arduino.

Each ESC is given power by the 12V battery. The Y connector is affixed to the battery and then each ESC is connected to the Y connector. The ESC connects to the motor through the 3-phase power connections. Each ESC connects to only one motor. Leave the battery unplugged until you are ready to start up the IPASS.

The three cameras are all connected to the USB hub. The hub is then plugged into the Gumstix's USB port. Pins 27 and 29 from the Gumstix need to be connected to A0 and A1 on the Arduino micro respectively.

Table 15: Connections in the IPASS electronics box

Connection 1	Type	Connection 2
Tobi Pin 27	Signal	Arduino A0
Tobi Pin 29	Signal	Arduino A1
ESC 1 Power	Power	Tobi Power
ESC 1 Signal	Signal	Arduino 9
ESC 1 Ground	Ground	Tobi Ground
ESC 2 Power	Power	Arduino Vcc
ESC 2 Signal	Signal	Arduino 9
ESC 2 Ground	Ground	Arduino Ground
Camera 1	Data	USB hub
Camera 2	Data	USB hub
Camera 3	Data	USB hub
USB hub	Data	Gumstix USB
Battery	Power	Y Connector
Y Connector 1	Power	ESC 1
Y Connector 2	Power	ESC 2
ESC 1 3-phase	Power/Signal	Motor 1
ESC 2 3-phase	Power/Signal	Motor 2

With all electronic connections established, each component can be attached to an internal wall of the Delrin electronics box. The Gumstix and the Arduino are mounted on opposite sides while the two ESCs are mounted on the remaining two opposite inner walls. Once two walls are placed, the internal Coroplast supports can be integrated into the box. Ensure that the Gumstix's antenna is mounted through the Coroplast supports' holes. The cameras are screwed into the camera mount which is then attached to the bottom of the walls. Insert the battery into the center of the support and leave power cables to the battery and Y-connector accessible. Place the nose cone on top of the electronics box and screw it into place.

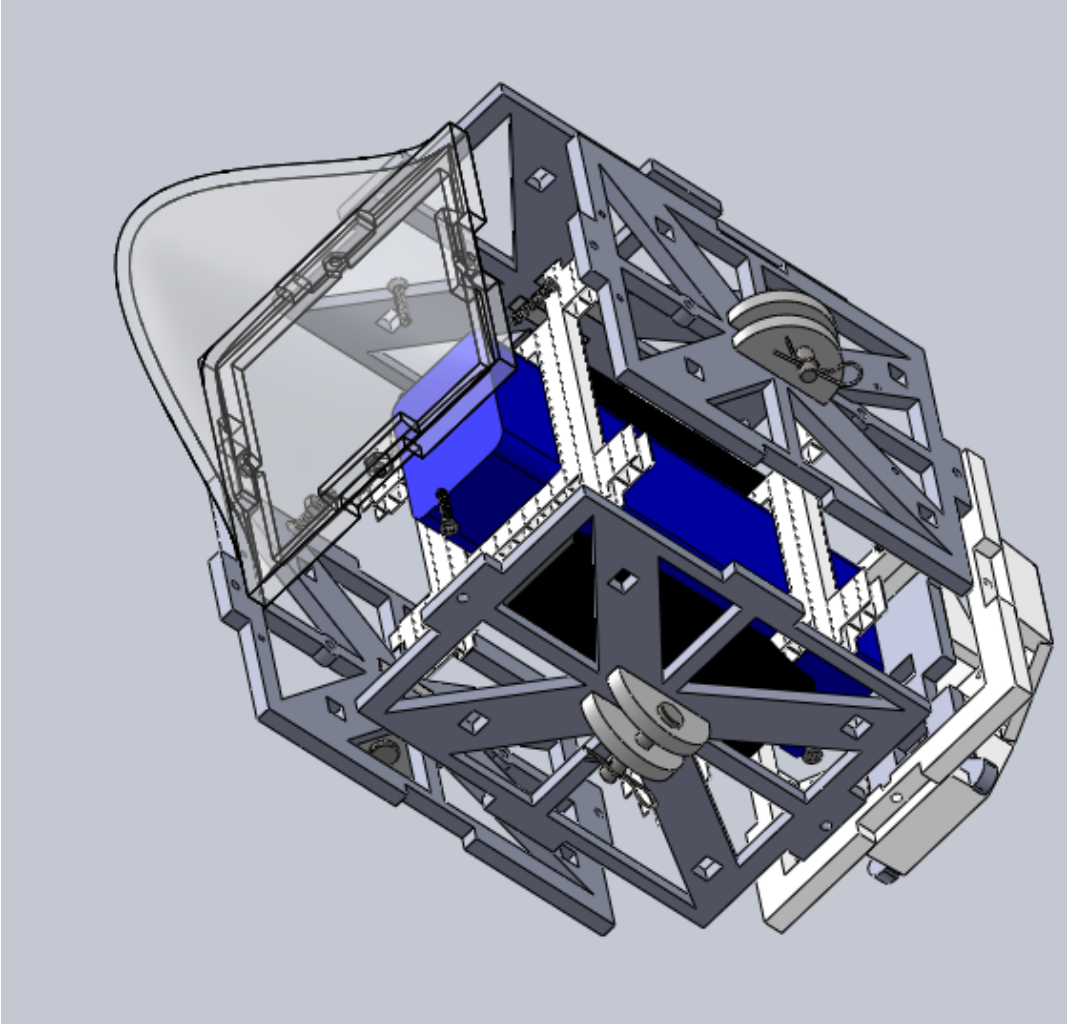


Figure 141: Exploded view of the electronics box. Electrical components have been hidden

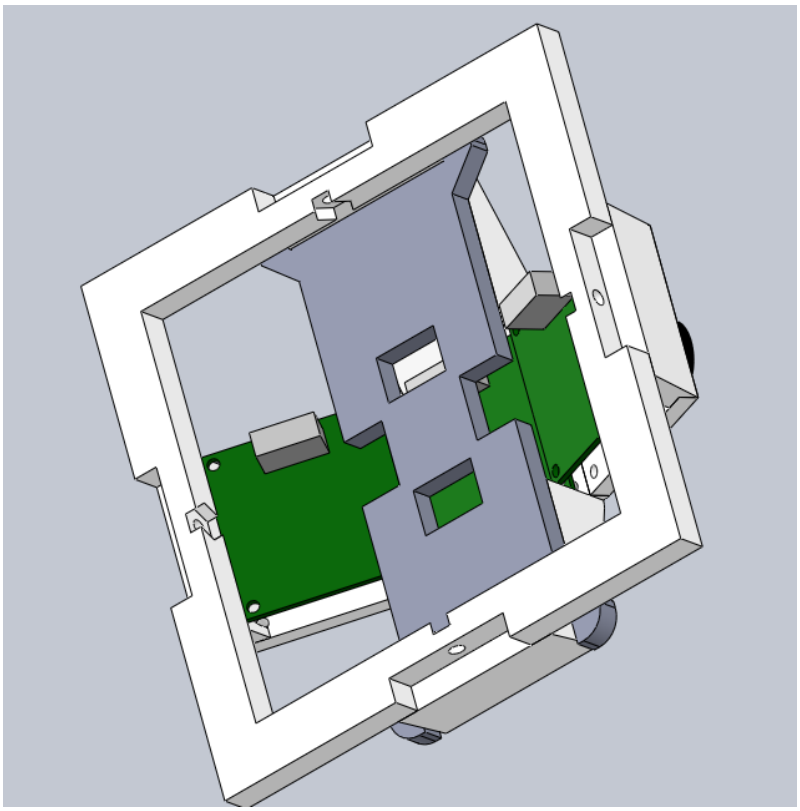


Figure 142: Inner view of the camera mount.

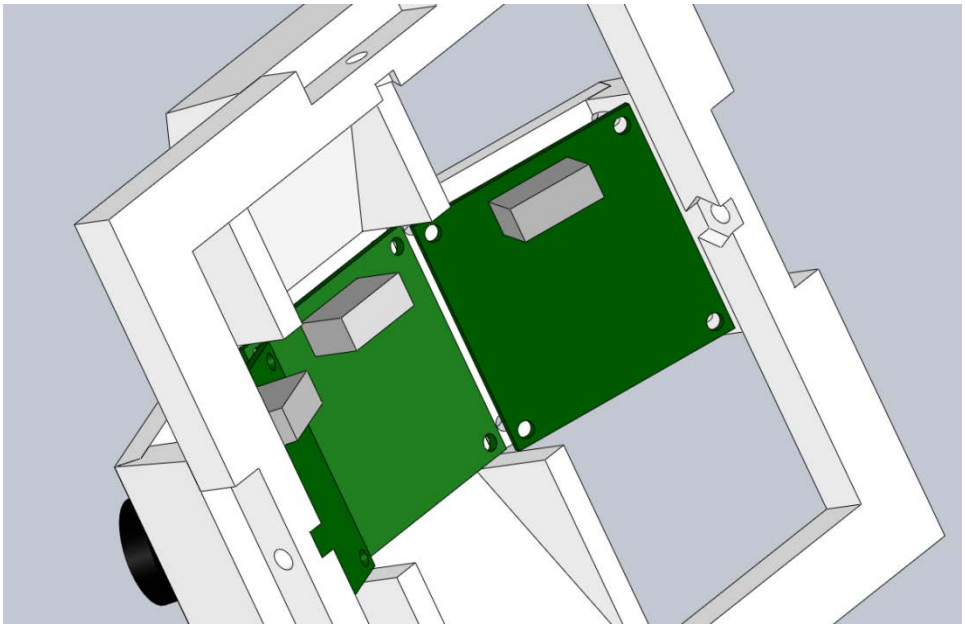


Figure 143: Reference view for camera placement and orientation.

Full Chassis

The IPASS is stored in separate parts within its Pelican case. These parts are as follows: 1 electronics box, 1 dual motor assembly with propellers, 4 carbon fiber rods, 4 quarter panels, and 2 support rings.

1. Insert the four quarter panels into their corresponding Delrin mounts and secure them with a clevis pins one at a time.

Figure 144: Affixing the panels to the electronic box with clevis pins

2. Insert the motor assembly, taking care to ensure that the propellers are oriented correctly (the text should be facing up).
3. Secure the motor mount with four zip ties around each of the connecting quarter panels.
4. Place both rings around the quarter panels to secure them in place.

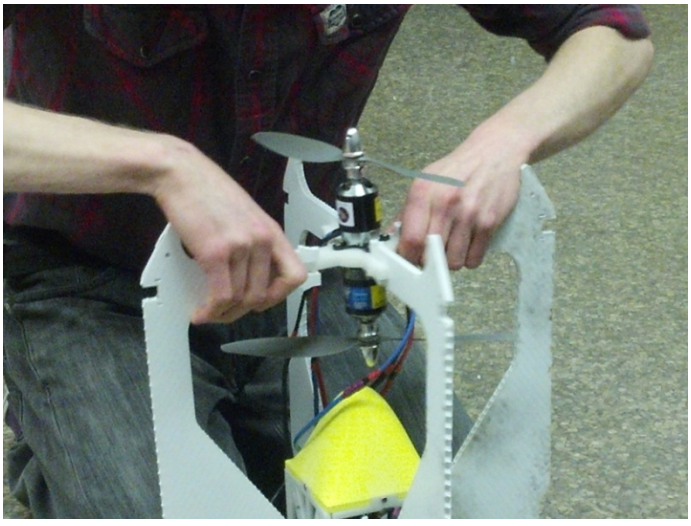


Figure 145: Affixing the motor mounting in place

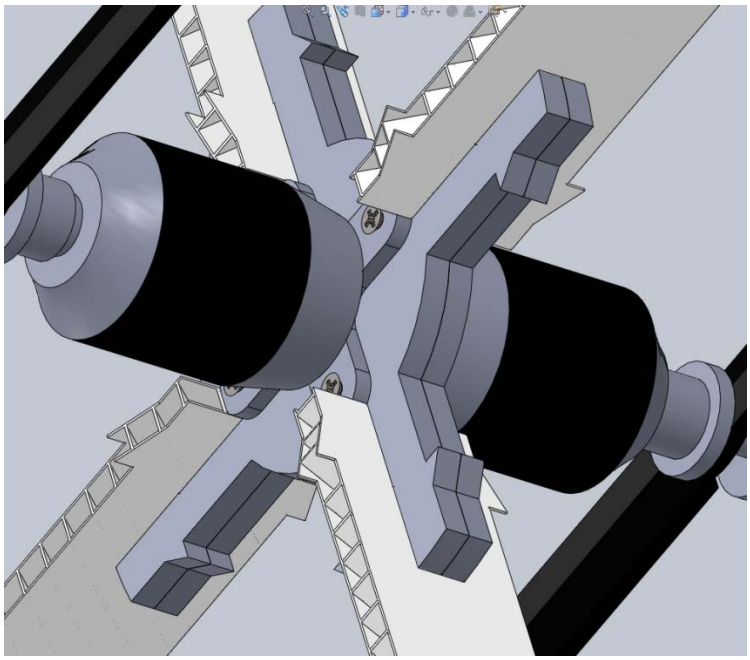


Figure 146: Detailed view of the motor mount

5. Attach the four carbon fiber rods the bottom ends of the quarter panels using two zip ties each.

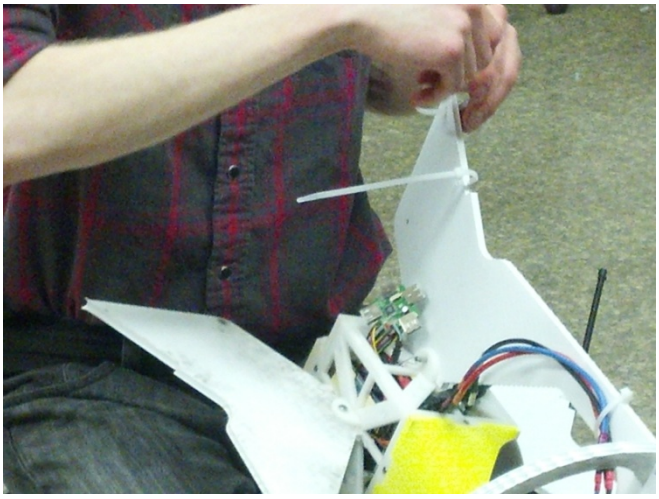


Figure 147: Affixing the zip ties to the chassis for the carbon fiber rods

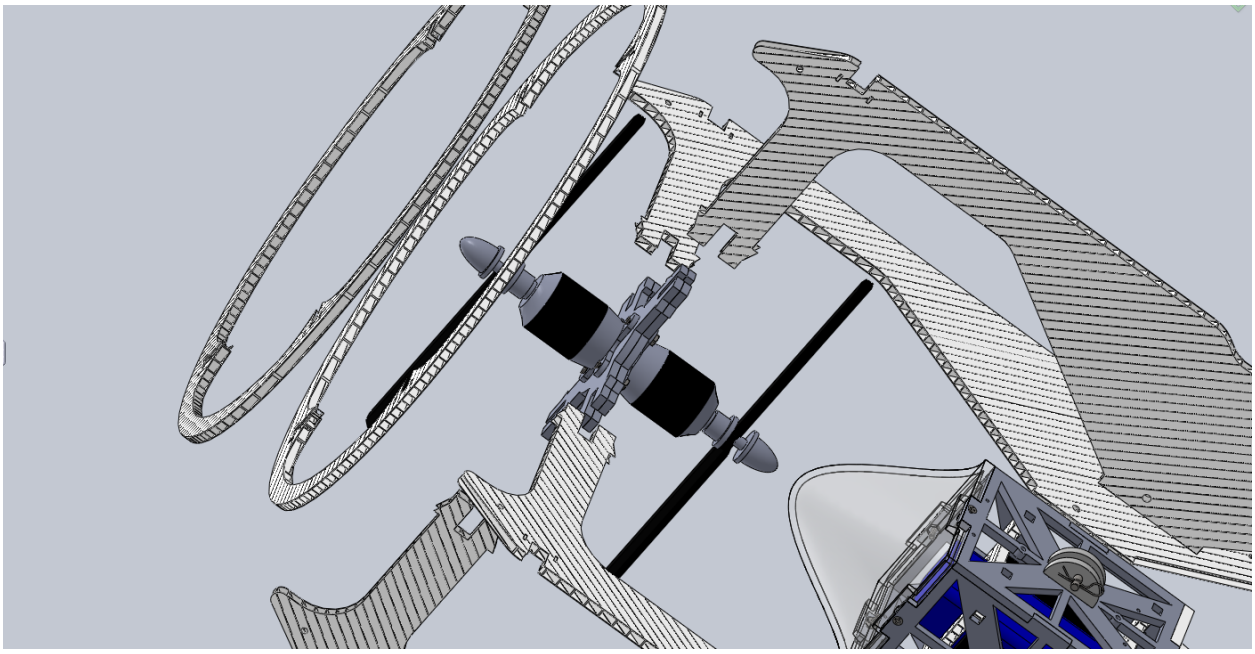


Figure 148: Exploded side view of the chassis

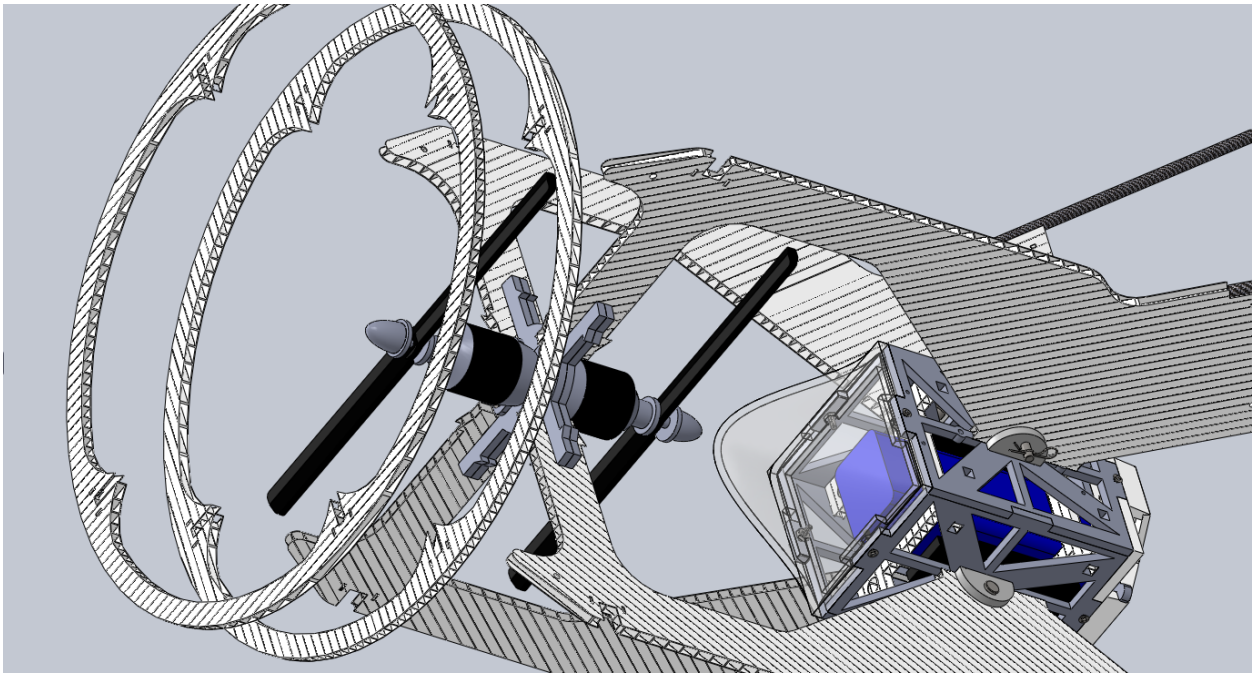


Figure 149: Exploded top view of the chassis

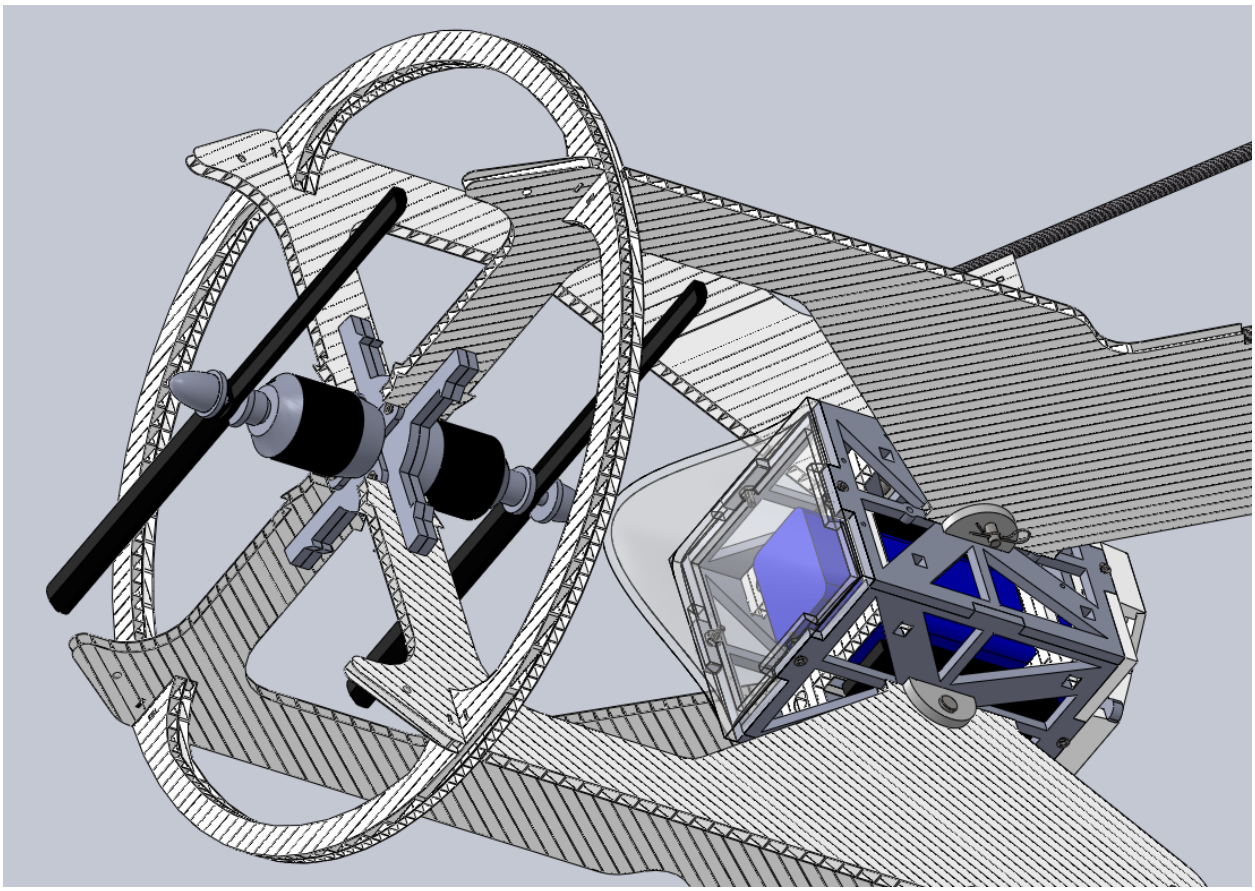


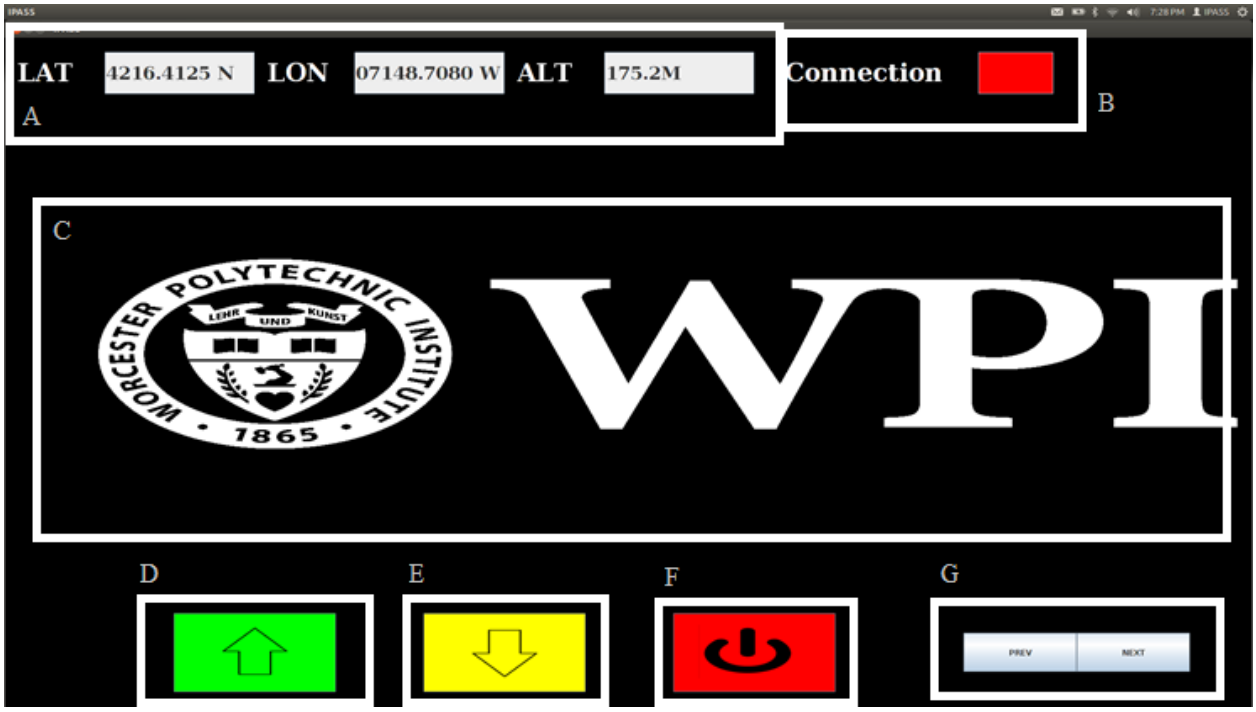
Figure 150: Completed chassis

Operation

Launch Procedure

1. Assemble the IPASS and stand it in a flat area
2. Remove the lens caps from the cameras
3. Connect the battery to turn on the IPASS. A red indicator light from the Arduino, blue and green lights from the Gumstix, and a short series of beeps from the motors indicates everything is booting up normally
4. Startup the IPASS GUI on the ground station a safe distance from the IPASS
5. Once the ground station has established connection with the IPASS, press the launch button to begin flight
6. Use the RC controller to control the flight of the IPASS while it captures image data
7. While the IPASS is in flight the land button can be pressed to immediately begin landing
8. While the IPASS is in flight the abort button can be pressed immediately shut down all systems in the IPASS
9. When a flight is complete, if the IPASS has been recovered disassemble it and replace its parts into the carrying case

GUI



GUI Diagram

A. *GPS Data:* From left to right: latitude, longitude, and altitude. Latitude and longitude values are in NMEA format. Altitude is in meters
B. *Connection Indicator*: Indicator is red when disconnected from IPASS. Indicator turns green when a connection is established

C. *Image Data*: displays stitched image data. Updates whenever a new set of images is received and stitched
D. *Launch Button*: Activates the IPASS allowing RC control.
E. *Land Button*: Causes the IPASS to land by reducing RC throttle
F. *Abort Button*: shuts down all systems
G. *Image History Buttons*: Allows scrolling through image data.

Appendix P: List of Acronyms

Acronym	Meaning
ABS	Acrylonitrile Butadiene Styrene
AFRL	Air Force Research Laboratories
ARM	Advanced RISC Machines
DoD	Department of Defense
DOF	Degrees of Freedom
ESC	Electronic Speed Controller
FAA	Federal Aviation Administration
GPS	Global Positioning System
GSD	Ground Sampling Distance
GUI	Graphical User Interface
IMU	Inertial Measurement Unit
IPASS	Intelligent Portable Aerial Surveillance System
ISR	Intelligence, Surveillance, and Reconnaissance
LiFePo	Lithium Iron Phosphate (battery)
Li-Po	Lithium Polymer (battery)
MAV	Micro Air Vehicles
NMEA	National Marine Electronics Association
OAI	Ohio Aerospace Institute
PPM	Pulse Period Modulation
RC	Remote Control
RANSAC	Random Sample Consensus
SIFT	Scale Invariant Feature Transform
SPI	Serial Peripheral Interface
SURF	Speeded-Up Robust Features
TCP	Transmission Control Protocol
UAV	Unmanned Aerial Vehicle
VTOL	Vertical Take-Off and Landing
WPI	Worcester Polytechnic Institute
WSU	Wright State University